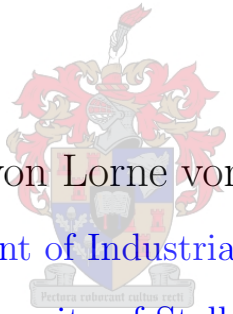
Integration of ranking and selection methods with the multi-objective optimisation cross-entropy method



Chantel von Lorne von Saint Ange

Department of Industrial Engineering

University of Stellenbosch

Supervisor: Professor J Bekker

Thesis presented in partial fulfilment of the requirements for the degree of Master of Engineering in the Faculty of Engineering at Stellenbosch University

M. Eng Industrial

March 2015

Declaration

By submitting this thesis electronically, I declare that the entirety of the work contained therein is my own, original work, that I am the sole author thereof (save to the extent explicitly otherwise stated), that reproduction and publication thereof by Stellenbosch University will not infringe on any third party rights and that I have not previously in its entirety or in part submitted it for obtaining any qualification.

Date: 1 December 2014

Acknowledgements

I would like to express my sincere gratitude to the following people and organisations

- James Bekker for his guidance, patience and sense of humour.
- My parents, Jeannine and Eberhard, for giving me the opportunity to become an Industrial Engineer and encouraging me to study further.
- My sister Nicole and my grandparents for always being there for me.
- Craig Campbell for his unconditional love and support.
- All the friends I have made during my time at Stellenbosch University for the wonderful memories.
- All of the students in the *Unit for Systems Modelling and Analysis* for the support and sharing my passion.

Abstract

A method for multi-objective optimisation using the cross-entropy method (MOO CEM) was recently developed by Bekker & Aldrich (2010) and Bekker (2012). The method aims to identify the non-dominated solutions of multi-objective problems, which are often dynamic and stochastic. The method does not use a statistical ranking and selection technique to account for the stochastic nature of the problems it solves. The research in this thesis aims to investigate possible techniques that can be incorporated into the MOO CEM.

The cross-entropy method for single-objective optimisation is studied first. It is applied to an interesting problem in the soil sciences and water management domain. The purpose of this was for the researcher to grasp the fundamentals of the cross-entropy method, which will be needed later in the study.

The second part of the study documents an overview of multi-objective ranking and selection methods found in literature. The first method covered is the multi-objective optimal computing budget allocation algorithm. The second method extends upon the first to include the concept of an indifference-zone. Both methods aim to maximise the probability of correctly selecting the non-dominated scenarios, while intelligently allocating simulation replications to minimise required sample sizes. These techniques are applied to two problems that are represented by simulation models, namely the buffer allocation problem and a classic single-commodity inventory problem. Performance is measured using the hyperarea indicator and Mann-Whitney U-tests. It was found that the two techniques have significantly different performances, although this could be due to the different number of solutions in the Pareto set.

In the third part of the document, the aforementioned multi-objective ranking and selection techniques are incorporated into the MOO CEM. Once again, the buffer allocation problem and the inventory problem were chosen as test problems. The results were compared to experiments where the MOO CEM without ranking and selection was used. Results show that the MOO CEM with ranking and selection has various affects on different problems. Investigating the possibility of incorporating ranking and selection differently in the MOO CEM is recommended as future research. Additionally, the combined algorithm should be tested on more stochastic problems.

Opsomming

'n Metode vir meerdoelige optimering wat gebruik maak van die kruis-entropie-metode (MOO CEM) is onlangs deur Bekker & Aldrich (2010) en Bekker (2012) ontwikkel. Die metode mik om die nie-gedomineerde oplossings van meerdoelige probleme te identifiseer, wat dikwels dinamies en stogasties is. Die metode maak nie gebruik van 'n statistiese orden-en-kies tegniek om die stogastiese aard van die problem aan te spreek nie. Die navorsing in hierdie tesis poog om moontlike tegnieke wat in die MOO CEM opgeneem kan word, te ondersoek.

Die kruis-entropie-metode vir enkeldoelwit optimering is eerste bestudeer. Dit is toegepas op 'n interessante probleem in die grondwetenskappe en waterbestuur domein. Die doel hiervan was om die navorser die grondbeginsels van die kruis-entropie metode te help verstaan, wat later in die studie benodig sal word.

Die tweede gedeelte van die studie verskaf 'n oorsig van meerdoelige orden-en-kies metodes wat in die literatuur aangetref word. Die eerste metode wat bespreek word, is die optimale toedeling van rekenaarbegroting vir multi-doelwit optimering algoritme. Die tweede metode brei uit oor die eerste metode wat die konsep van 'n neutrale sone insluit. Beide metodes streef daarna om die waarskynlikheid dat die nie-gedomineerde oplossings korrek gekies word te maksimeer, terwyl dit ook steekproefgroottes probeer minimeer deur die aantal simulasieherhalings intelligent toe te ken. Hierdie tegnieke word toegepas op twee probleme wat verteenwoordig word deur simulasiemodelle, naamlik die buffer-toedelingsprobleem en 'n klassieke enkelitem voorraadprobleem. Die prestasie van die algoritmes word deur middel van die hiperarea-aanwyser en Mann Whitney U-toetse gemeet. Daar is gevind dat die twee tegnieke aansienlik verskillend presteer, alhoewel

dit as gevolg van die verskillende aantal oplossings in die Pareto versameling kan wees.

In die derde gedeelte van die dokument, is die bogenoemde meerdoelige orden-en-kies tegnieke in die MOO CEM geïnkorporeer. Weereens is die buffer-toedelingsprobleem en die voorraadprobleem as toetsprobleme gekies. Die resultate was met die eksperimente waar die MOO CEM sonder orden-en-kies gebruik is, vergelyk. Resultate toon dat vir verskillende probleme, tree die MOO CEM met orden-en-kies anders op. 'n Ondersoek oor 'n alternatiewe manier om orden-en-kies met die MOO CEM te integreer is as toekomstige navorsing voorgestel. Bykomend moet die gekombineerde algoritme op meer stogastiese probleme getoets word.

Contents

List of Figures

List of Tables

Nomenclature

Acronyms

API	Application programming interface
BAP	Buffer allocation problem
CEM	Cross-entropy method
CID	Crop identification
CS	Correct selection
EA	Equal allocation
HA	Hyperarea
HRU	Hydrological response unit
IZ	Indifference-zone
JAMS	Jena Adaptable Modelling System
LB	Lower bound
MAUT	Multi-attribute utility theory
MOCBA_IZ	Multi-objective optimal computing budget allocation algorithm with indifference-zone framework
MOCBA	Multi-objective optimal computing budget allocation
MOO CEM	Multi-objective optimisation using the cross-entropy method
MOO	Multi-objective optimisation
MOP	Multi-objective problem
MORS	Multi-objective ranking and selection
OCBA	Optimal computing budget allocation
RID	Crop rotation identification

R&S	Ranking and selection
SRSIP	Sequential ranking and selection of an incomplete Pareto set
TOA	Theoretical optimal allocation
TSPS	Two-stage-Pareto-set-selection procedure
UB	Upper bound
UCBA	Uniform computing budget allocation
WIP	Work-in-progress
ZAR	South African Rands

Greek Symbols

α	Smoothing parameter for the cross-entropy method
α_i	Proportion of the total simulation budget allocated to scenario i
Δ	Number of replications to allocate per iteration of MOCBA
δ	Termination counter of the cross-entropy method
ε^*	Error limit for the Type I and Type II errors
ϵ_c	MOO CEM common termination threshold
γ	Cross-entropy optimisation rare-event threshold value
δ_k^*	Indifference-zone for objective k
λ_i	Mean exponential failure rate for machine i
μ	Mean of a distribution
Ω	Feasible region of an optimisation problem
ω_i	Additional number of simulation replications for scenario i
ϕ	Mathematical function, including probability mass and density function
ψ_i	Performance index of scenario i
ρ_E	Ranking threshold for the Pareto ranking algorithm
σ	Standard deviation of a distribution
τ	Maximum number of replications that can be allocated to a scenario at each iteration

θ	Mathematical function, including probability mass and density function
ϱ	User-specified rare-event threshold value for the cross-entropy method

Roman Symbols

B_i	Size of buffer space i
D	Number of decision variables in an optimisation problem
e_1	Type I error
e_2	Type II error
E_i	Event that scenario i is non-dominated by all other scenarios
f	Mathematical function, including probability mass and density function
I	Indicator function
j_i	Index of the scenario that is most likely to dominate scenario i
H	Number of objectives in an optimisation problem
$k^i_{j_i}$	Index of the objective of j_i that dominates the corresponding objective of scenario i with the lowest probability
l	Rare-event probability in importance sampling
l_l	RID lower limit
l_u	RID upper limit
M	Number of inequality constraints
m	Number of machines in the buffer allocation problem
n	Number of buffers in the buffer allocation problem
n_d	Number of elements in the discrete decision vector
N	Number of scenarios under investigation in MORS or population size for population based algorithms
n_0	Number of initial replications to perform for each scenario
P^*	Minimum acceptable probability of correctly selecting the best scenario for TSPS

p_h	Probability of inverting MOO CEM histogram counts
Q	Number of equality constraints
R_i	Number of simulation replications for scenario i
r_i	Mean exponential repair rate for machine i
R_T	Total computing budget or number of replications
s	Reorder level
S	Reorder quantity
S^A	Subset A of $\mathcal{S}$
S^B	Subset B of $\mathcal{S}$
S_{IP}	Incomplete Pareto set
S_p	Approximated Pareto set
S_p^A	Subset A of S_p
S_p^B	Subset B of S_p
$\bar{S}_p$	Approximated non-Pareto set
T_R	Measure of throughput rate
U	Random number, uniformly distributed on $(0, 1)$
W_P	Measure of work-in-progress

Other Symbols

$\mathcal{D}$	Kullback-Leibler distance
$\mathbb{E}$	Mathematical expectation
$\mathcal{H}$	The set of H objectives
$\mathcal{P}^*$	Pareto optimal set
$\mathcal{P}_T^*$	True Pareto front
$\mathcal{S}$	Design space for MORS problems containing all N competing scenarios
$\mathcal{V}$	Parameter vector set of the cross-entropy method
$\mathbf{W}$	Working matrix of the Pareto ranking algorithm
$\mathcal{X}$	Feasible region of cross-entropy optimisation problem

Chapter 1

Introduction

This chapter serves as an introduction to the research presented in this thesis. The background to the research is presented, followed by the problem statement and the research methodology.

1.1 Background

Systems that are dynamic and complex in nature often have no closed form analytical solutions and are usually modelled using discrete-event computer simulation. The systems studied in this research are of this type. The *simulation model* is used to evaluate the performance of various scenarios of the system. Each possible *scenario* is made up of one or more *decision variables*, which could be parameter values or a physical composition of the system.

When a very large number of possible combinations of decision variable values exist and the goal is to determine the best combination, an optimisation algorithm can be integrated with the simulation model. The best scenario is defined in terms of the maximum or minimum expected performance of some measure of the system. Simulation can only evaluate the performance of a small, finite set of scenarios, whereas optimisation searches for near-optimal values of decision variable values, in a large decision space, that would optimise the performance of the system. Integrating the two concepts is known as *simulation optimisation*.

To ensure success in determining the best scenario(s), simulation optimisation algorithms are required to strike a balance between exploring the solution

space for new, improved solutions, while exploiting the already found solutions to determine how good they actually are. This trade-off is known as search vs. selection or *exploration* vs. *exploitation*.

When a simulation model is not subject to uncertainty, it is *deterministic*. In deterministic models, a certain set of input quantities and relationships, always results in the same simulation output. *Stochastic* simulation models have probabilistic components and thus the output thereof can be used to estimate the true characteristics of the system. Typically, the expected values of performance measures are estimated via point estimators (Law & Kelton, 2000). In this case, several pseudo-independent *replications* have to be run, for the same input parameters, to control the statistical estimation error. Thus several samples or observations are taken for the scenario.

In some simulation models, a replication may take a long time to execute, and obtaining estimates from several replications becomes time-consuming and computationally expensive. When optimisation is applied to the problem, several replications are required, per combination of decision variable values. In addition, the optimisation algorithm typically performs a number of iterations in search of the optimal decision variable values. This becomes a computational burden and algorithmic efficiency becomes a major focus of interest in this context.

Efficiency in simulation optimisation means obtaining quality results, with minimal effort. To enhance the efficiency of deterministic models, one can either develop more efficient simulation technology or use better computers, to reduce the simulation time of experiments (Chen *et al.*, 1996).

In the case of stochastic models, *ranking and selection* procedures can improve simulation optimisation by minimising the number of simulation replications, while ensuring the best scenario is identified, with a certain level of confidence. There are two main approaches to ranking and selection, namely: *indifference-zone* methods and the *optimal computing budget allocation* framework. The formulations differ by whether the requirement is imposed on the evidence of correct selection, or on the simulation budget.

If a ranking and selection procedure is not employed in simulation optimisation, then usually a small, fixed number of simulation replications are allocated to each scenario. In this case, the true stochastic nature of the problem is not

always captured. Consequently, either the simulation replications are too few to estimate the performance measure sufficiently, or the simulation replications are too many to be computationally efficient (Lee *et al.*, 2008).

Ranking and selection procedures compare a finite and relatively small number of scenarios, so that all the scenarios can be evaluated. If there is a very large search space, a need arises to integrate statistical ranking and selection with a search algorithm.

In the case of a problem with more than one performance measure, multi-objective optimisation and *multi-objective* ranking and selection methods need to be applied. A specific metaheuristic to solve multi-objective optimisation problems is considered for this study.

1.2 Problem statement

In a recent endeavour, the *cross-entropy method* for single-objective optimisation was adapted to be applied to multi-objective, stochastic cases (Bekker, 2012; Bekker & Aldrich, 2010). The new method uses a Pareto-based ranking algorithm to determine the best combinations of objective function values. It is then up to the decision maker to choose their preferred solution from the set, by means of a post-analysis process.

The problem is that the Pareto ranking algorithm compares the numerical objective function values of scenarios, based on a small, equal number of simulation replications. It is therefore unknown if the scenarios have been sufficiently evaluated for randomness in the model, so as to accurately select the top performing alternative. Moreover, inaccurate solutions may mislead the search algorithm, so performing statistical output analysis may be beneficial for both search efficiency and simulation efficiency.

Based on the aforementioned reasons, the research task is:

To investigate the ranking of the multi-objective optimisation method using the cross-entropy method algorithm.

1.3 Methodology

At an early stage in the research process, the researcher came across a practical problem from a water management workshop, hosted by the Council for Scientific and Industrial Research, and the Water Institute of Stellenbosch University. It was found that the cross-entropy method for single-objective optimisation could be used to solve the water management problem. The researcher seized this opportunity to apply the cross-entropy method to a real-life problem, in order to learn the method, through practical implementation. This forms the first part of this study.

The water management problem entails optimally assigning land uses to pieces of land in a water catchment area in the Western Cape Province. The objective of the optimisation problem is to minimise the salinity levels of the water in the catchment. These levels have been increasing during the last century, which has led to a deteriorating water quality as well as reduced the fertility of landscapes. The optimisation model built interacts with an existing hydrological model (3rd party) that simulates the hydrological behaviour in the water catchment. The hydrological model is deterministic and acts as a black-box. Water catchment managers as well as farmers can use the results of the study as a guideline when forming land use regulations and a dryland salinity management strategy for the area.

The second part of the study moves onto stochastic multi-objective simulation models. Specifically, the aim is to investigate the field of *multi-objective ranking and selection*. Literature on ranking and selection was reviewed, to find approaches that take the stochastic nature of simulation output data into account, whilst using a Pareto approach. The methods identified are applied to two case studies and the performance of the algorithms is compared. The case

studies are the buffer allocation problem and the (s, S) inventory model, and are built in Simio (3rd party).

Once the field of multi-objective ranking and selection has been studied, and conclusions drawn on solution methods, the research progresses to the third and final part of the study. The aim is to incorporate multi-objective ranking and selection into the *multi-objective optimisation method using the cross-entropy method*. Both the *multi-objective optimal computing budget allocation* algorithm and the *multi-objective optimal computing budget allocation with the indifference-zone* algorithm are incorporated in the multi-objective optimisation method using the cross-entropy method. Two multi-objective optimisation problems, from Bekker (2012) are experimented on, so that equal comparisons can be made with regard to the multi-objective optimisation method using the cross-entropy method, with and without ranking and selection.

1.4 Structure of the document

This chapter serves to introduce simulation optimisation and the different types of models that it can be applied to. Considering the multi-objective optimisation method using the cross-entropy method, the problem statement is developed. The methodology for the research is then discussed. The structure of the document is formed by the methodology.

The water management case study is presented in **Chapter 2**. This begins with a brief overview of the cross-entropy method and literature study on salinity in the water catchment and the hydrological model. This is followed by a description of the specific problem. The optimisation model is then formulated and the experimentation is documented. The chapter concludes with the results of the water management case.

In **Chapter 3**, an introduction to ranking and selection is given, followed by an in depth literature review on the multi-objective ranking and selection methods relevant to this study. The experimental design for the experiments performed for multi-objective ranking and selection is presented. This includes a description of two test problems, literature on applicable performance indicators

and significance testing, and the parameter settings of the algorithms. Finally, a summary and discussion of the experimental results is given.

An overview on general integration of statistical selection with search algorithms is presented in **Chapter 4**. The chapter also contains the description of the multi-objective optimisation method using the cross-entropy method. The experimentation procedure for this part of the research is documented, followed by the results.

The summary and general conclusions of the research are presented in **Chapter 5**.

Appendix A contains the basic Matlab® code for the simple multi-objective optimal computing budget allocation method. An extension of a Pareto based ranking algorithm to include the concept of an indifference-zone is provided in **Appendix B**. Results from testing various parameter values for two multi-objective ranking and selection procedures, on two simulations models, are included in **Appendix C**.

Chapter 2

Water management optimisation study using the cross-entropy method

This chapter provides a theoretical overview of the cross-entropy method (CEM) as well as the problem formulation in an optimisation context. The CEM is applied to the water management optimisation problem. The CEM is used to find optimal solutions from the large decision space, where brute-force methods are infeasible. The background to dryland salinity and the study area is presented, followed by the hydrological model, used to obtain the objective function value. The chapter then delves into the current method of selecting crops, highlighting the need for an optimisation model. The optimisation model is then developed and experiments are performed. Finally, the experimental results are presented, along with conclusions and suggested future work.

2.1 The cross-entropy method for single-objective optimisation

The CEM is a technique applied to optimisation problems and rare event simulation. It was developed by Reuven Rubinstein (Rubinstein, 1999) and has its foundations in importance sampling and the Kullback-Leibler cross-entropy.

For this study the CEM is reviewed from an optimisation perspective, based on Rubinstein & Kroese (2004). For more information and numerous applications of the CEM the reader is referred to Kroese & Rubinstein (2005); Rubinstein & Kroese (2004).

The CEM is an iterative method where each iteration consists of two stages. Every decision variable domain is associated with a probability density function. In the first stage, a value for each decision variable is drawn from the probability densities. In other words, a sample is generated. In the second stage, the parameters of the distribution are updated based on the output of the objective function. This attempts to increase the likelihood of an improved sample in the next iteration. The second phase entails minimising the Kullback-Leibler divergence or cross-entropy distance, from which the method acquires its name (Rubinstein & Kroese, 2004).

As the CEM is to be applied to a discrete parameter water management problem, discussion of the CEM is focused on discrete optimisation.

Let $\mathcal{X}$ be a finite set of states (decision variables) and f be the objective or performance function on $\mathcal{X}$. Considering a maximisation problem, the aim is to determine the maximum of f over $\mathcal{X}$ and the corresponding states at this specific maximum (γ^*)

$$f(\mathbf{x}^*) = \gamma^* = \max_{\mathbf{x}\in\mathcal{X}} f(\mathbf{x}). \tag{2.1}$$

To solve the optimisation problem in (2.1), the CEM requires that an estimation problem be associated with it. For this to occur, a collection of indicator functions $\{I_{\{f(\mathbf{x})\geq\gamma\}}\}$ on $\mathcal{X}$ for different values $\gamma \in \mathbb{R}$ are defined. Furthermore, let $\{\phi(\cdot, \mathbf{v}), \mathbf{v} \in \mathcal{V}\}$ be a family of discrete probability mass functions on $\mathcal{X}$, where $\mathbf{v}$ is a real-valued parameter vector. Suppose $\mathbf{u} \in \mathcal{V}$, and associate with (2.1) the problem of estimating the probability that $f(\mathbf{X})$ is greater than or equal to a level (real number) γ. This is given by

$$l = P_{\mathbf{u}}\{f(\mathbf{X}) \geq \gamma\} = \sum_{\mathbf{x}} I_{\{f(\mathbf{x})\geq\gamma\}}\phi(\mathbf{x};\mathbf{u}) = \mathbb{E}_{\mathbf{u}} I_{\{f(\mathbf{X})\geq\gamma\}}, \tag{2.2}$$

where $\mathbb{E}_{\mathbf{u}}$ is the corresponding mathematical expectation. If this probability is very small, $f(\mathbf{X}) \geq \gamma$ is a rare event. Importance sampling can be used to

estimate l. This is achieved by taking random samples on $\mathcal{X}$ from a different mass function θ and estimating l with the likelihood ratio estimator

$$\hat{l} = \frac{1}{N} \sum_{k=1}^{N} I_{\{f(\mathbf{X}_k) \geq \gamma\}} \frac{\phi(\mathbf{X}_k, \mathbf{u})}{\theta(\mathbf{X}_k)}. \tag{2.3}$$

The best approach to estimate l is to use the change of measure with mass function

$$\theta^*(\mathbf{x}) = \frac{I_{\{f(\mathbf{x}) \geq \gamma\}} \phi(\mathbf{x}; \mathbf{u})}{l}, \tag{2.4}$$

which yields the probability

$$l = \frac{I_{\{f(\mathbf{X}_k) \geq \gamma\}} \phi(\mathbf{X}_k; \mathbf{u})}{\theta^*(\mathbf{X}_k)}. \tag{2.5}$$

Since the value of θ^* depends on the unknown parameter l, another way to approximate θ^* is by choosing it from the family of mass functions $\{\phi(\cdot, \mathbf{v})\}$. In this case, $\mathbf{v}$ is the reference parameter and it is chosen such that the distance between θ^* and $\{\phi(\cdot, \mathbf{v})\}$ is minimised. The distance between the two mass functions is known as the Kullback-Leibler distance or the cross-entropy. It is expressed as

$$\begin{aligned} \mathcal{D}(\theta, \phi) &= \mathbb{E}_\theta \left[\log \frac{\theta(\mathbf{X})}{\phi(\mathbf{X})} \right] \\ &= \sum_{\mathbf{x}} \theta(\mathbf{x}) \log \frac{\theta(\mathbf{x})}{\phi(\mathbf{x})} \\ &= \sum_{\mathbf{x}} \theta(\mathbf{x}) \log \theta(\mathbf{x}) - \sum_{\mathbf{x}} \theta(\mathbf{x}) \log \phi(\mathbf{x}). \end{aligned} \tag{2.6}$$

The likelihood estimator in (2.3) has the reference parameter

$$\mathbf{v}^* = \arg\max_{v} \frac{1}{N} \sum_{k=1}^{N} I_{\{f(\mathbf{X}_k) \geq \gamma\}} \ln \phi(\mathbf{X}_k, \mathbf{v}). \tag{2.7}$$

The concept for this is that if γ is has a value close to γ^*, most of the probability mass of $\phi(\cdot, \mathbf{v}^*)$ will be assigned near $\mathbf{x}^*$. Hereby, the distribution can be employed to generate an approximate solution.

In the discrete optimisation problem, one can draw observations for random vectors $\mathbf{X}_i = (X_{i1}, \ldots, X_{in_d})$, for $j = 1, \ldots, n_d$ elements in the decision vector. The estimator of p_j is

$$\hat{p}_j = \frac{\sum_{i=1}^{N} I_{\{\hat{f}(\mathbf{X}_i) \geq \gamma\}} I_{\{X_{ij}=j\}}}{\sum_{i=1}^{N} I_{\{\hat{f}(\mathbf{X}_i) \geq \gamma\}}}. \tag{2.8}$$

Instead of using (2.7) to update the parameter vector $\hat{P}_{t-1}$ to $\hat{P}_t$ directly, the algorithm implements a smoothed updating procedure

$$\hat{\mathbf{P}}_t = \alpha \tilde{\mathbf{P}}_t + (1 - \alpha)\hat{\mathbf{P}}_{t-1}, \tag{2.9}$$

where α is a smoothing constant which controls the convergence rate, so as not to prematurely converge.

The optimisation algorithm for the discrete case by Rubinstein & Kroese (2004) is shown as Algorithm 1.

Algorithm 1 Main CE Algorithm: discrete optimisation

1: Let the elements of $\hat{P}_0$ be a sample from $U(0,1)$. Set $t = 1$.
2: Generate a sample $\mathbf{X}_1, \ldots, \mathbf{X}_N$ using $\mathbf{P}_{t-1}$, and determine the sample $(1-\varrho)$-quantile $\hat{\gamma}_t$ of the performance function.
3: Using the same sample $\mathbf{X}_1, \ldots, \mathbf{X}_N$, update $\hat{p}_j$ with the expression in (2.8).
4: Smooth $\hat{\mathbf{P}}_t$ with (2.9).
5: If, for some $t \geq \delta$, say $\delta = 5$, $\hat{\gamma}_t = \hat{\gamma}_{t-1} = \ldots = \hat{\gamma}_{t-\delta}$, then stop; otherwise set $t \leftarrow t + 1$ and return to Step 2.

The CEM for the continuous case is similar to the discrete case and a description of it can be found in Rubinstein & Kroese (2004).

2.2 Background to the water management problem

This section presents an overview of important literature for the water management study. The concept and impacts of dryland salinity is described, followed by the details of the study area — the Sandspruit catchment. A catchment is the

area of land that is drained by a river and its tributaries. Finally, the hydrological simulator is presented.

2.2.1 Dryland salinity

Salinity refers to the salt concentration in soil or of a body of water. Salinity is quantified in terms of the electrical conductivity of the water measured, which possesses the units siemens per meter (S/m). Another approach to express salinity is gravimetrically — the mass of the total dissolved solids per volume of water. This is usually stated in grams/litre (g/l) and is also known as total dissolved salts (Richards, 1954).

Salt-affected soils occur in many regions of the world with varying magnitudes and properties. Briefly, the process of salinization is the accumulation of salts in the landscape to a stage that is detrimental to agricultural yield, environmental health and economic prosperity. It is a complex process that entails the movement of salts in water during seasonal cycles and the interaction of salts with groundwater (Rengasamy, 2006).

Salinization of land and water resources may either be classified as a natural occurrence (*primary salinity*) or induced by artificial processes (*secondary salinity*). Primary salinity is caused by the release of salts through the weathering of naturally saline rocks, the gradual withdrawal of an ocean and/or atmospheric deposition (Bugan *et al.*, 2012b). The latter process contributes salts of marine origin by aeolian (wind) and rainfall (Bugan, 2008). Secondary salinity is an outcome of human actions. It may either be a consequence of directly adding saline water, such as poor quality irrigation water and industrial waste, to soil and/or body of water, or it may be the effect of a change in a catchment's water balance, which causes salt stores to be mobilised (Bugan *et al.*, 2012b). The latter is known as dryland salinity which occurs in areas that are not irrigated (Bugan, 2008). A common anthropogenic activity that results in dryland salinity is the change of land use and land management strategies.

Researchers investigating dryland salinity impacts on Western Cape rivers found that the Berg River has been exhibiting an increasing trend in salinity levels (Fey & de Clercq, 2004). Further research was conducted to obtain more

knowledge about the salinity and water dynamics of the Berg River (see de Clercq *et al.* (2010)), due to its strategic importance in industrial and rural growth for the Western Cape (Fey & de Clercq, 2004). Other important reasons include the need to sustain in-stream ecology and the river acting as a source of freshwater in the Cape.

2.2.2 The Sandspruit catchment

The Sandspruit catchment is located near the town of Riebeek West in the Western Cape Province of South Africa (Figure 2.1). The area of the catchment is 152 km^2. According to Bugan *et al.* (2012a), the most common land uses in the catchment are cultivated lands and pastures. Wheat cultivation is the most common form of agriculture; with lupins, canola and grapes following. The Sandspruit River is a tributary of the Berg River (Bugan *et al.*, 2012a). This is relevant because the Berg River is an important source of freshwater in the Western Cape (Department of Water Affairs and Forestry, South Africa, 2004) and activities in the Sandspruit catchment affect the water quality and volume of the Berg River.

In a study by de Clercq *et al.* (2010), it was found that dryland salinity within the Sandspruit catchment was extensive. The increase in salinity is a result of naturally occurring saline geology and land use change, over more than a century, from indigenous vegetation to agricultural use. In addition to the mobilization of stored salt caused by land use change, because the catchment is in a semi-arid region its capacity to drain salt and water is limited, causing salt to build-up in the resources (Bugan *et al.*, 2012b).

According to de Clercq *et al.* (2010), the impact of dryland salinization in the catchment is a deteriorating water quality as well as a reduction of the fertility of the landscapes, affecting the agricultural activities (Bugan *et al.*, 2012b), water supply and ecology of the river system (de Clercq *et al.*, 2010). Consequently, the water in the catchment is not suitable for human consumption and is also not recommended for agricultural and/or industrial use (de Clercq *et al.*, 2010). Irrigating using this water could lead to crop loss and additional land degradation. In addition, the poor water quality could result in considerable economic losses and water supply problems (Bugan *et al.*, 2012b).

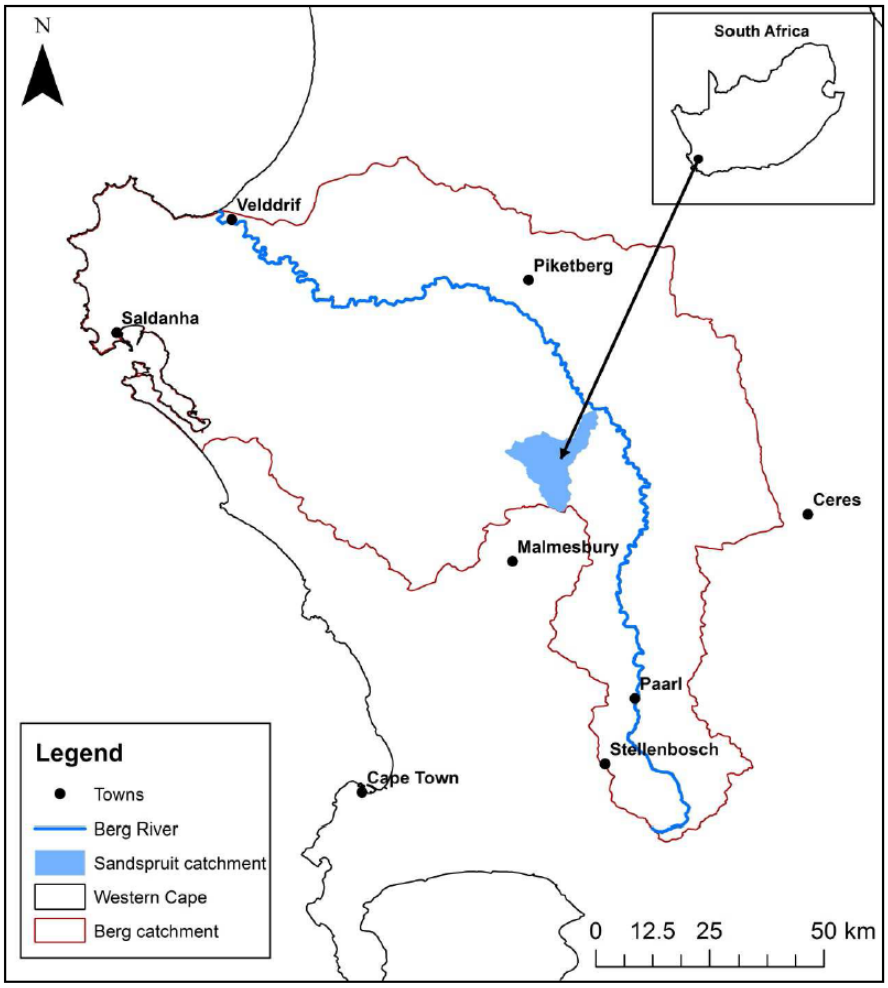


Figure 2.1: Location of the Sandspruit catchment in the Western Cape.

For these reasons the specific hydrological drivers, the causes and dynamics of salinization in the catchment, need to be identified and quantified in order to develop a dryland salinity management strategy for the catchment (Bugan *et al.*, 2012a). This will be completed with the aid of hydrological modelling, as it has been identified as a tool for the successful planning and operation of a catchment and the development of salinity management strategies (Bugan *et al.*, 2012b).

Researchers at the Department of Soil Science, Stellenbosch University and the Hydrosciences Group at the Council for Scientific and Industrial Research have been monitoring and collecting hydrological data in the Sandspruit catchment for

the past few years. This data was used to set up the JAMS/J2000-NaCl model. The scientists experimented on this medium-scale catchment, in the hope to use the results to predict consequences and make informed management decisions for the whole Berg River catchment (see Figure 2.1) (de Clercq *et al.*, 2010).

2.2.3 The JAMS/J2000 hydrological model and hydrological response units

In water management, hydrological modelling is a tool used to represent the water cycle of an area based on characteristics such as soil type, tillage, crops, precipitation and evapotranspiration, to name a few (de Clercq *et al.*, 2013). Hydrological models determine the water balance in a catchment, which is the relationship between all the components of the hydrological cycle in the catchment. It is a challenging and important topic in the hydrology field and becomes even more critical under the effects of human-induced land use change (Bugan *et al.*, 2012a).

The J2000 model is a process oriented hydrological modelling tool used to simulate the water balance and hydrological behaviour in large river catchments (Krause, 2002). The J2000 model contributes the process knowledge needed for the Jena Adaptable Modelling System (JAMS), a generalised framework implemented in Java for the development and application of environmental model components (Krause & Kralisch, 2005). For more information on JAMS and the J2000 model, the reader is referred to two webpages [1,2]. In a recent endeavour by Bugan (2014), a hydrological process module for salinity in river basins was added to the JAMS/J2000 model to form the JAMS/J2000-NaCl hydrological model. The module simulates water and inorganic salt fluxes, and land use at a catchment scale. The NaCl (in the model's name) represents the chemical formula of sodium chloride — the primary salt responsible for influencing salinity (Chapman, 1966). This model was used to simulate the hydrological behaviour of the Sandspruit catchment (Bugan *et al.*, 2012a).

The J2000 model subdivides a water catchment into Hydrological response units (HRUs) which are the modelling entities for the simulation (Fluegel, 1995).

[1] http://jams.uni-jena.de/

[2] http://jams.uni-jena.de/ilmswiki/index.php/Hydrological_Model_J2000

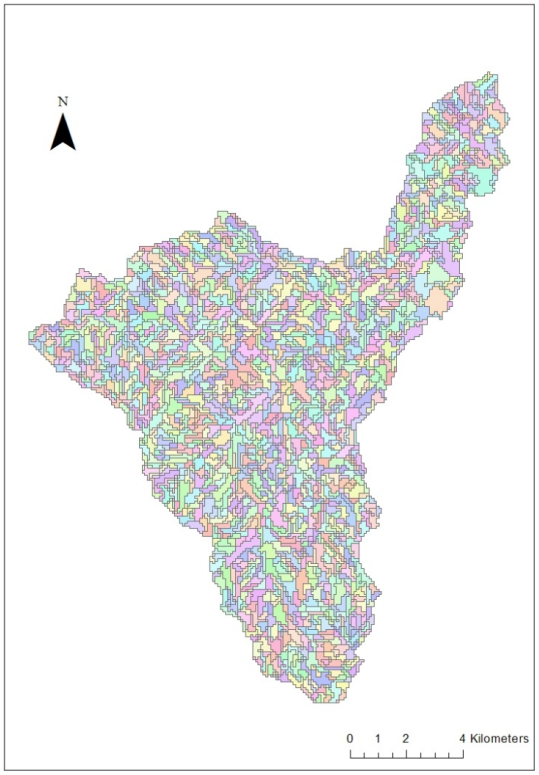


Figure 2.2: HRU delineation for the Sandspruit catchment.

The model makes use of a GIS (Geographic Information System) platform to delineate the HRUs according to spatial data of topography, aspect, soil, geology, land use and climate such that each HRU is uniform in its conditions (Krause, 2002). As a result the hydrological response within a HRU is similar, having a small variation of the hydrological process dynamics when compared to the dynamics of another HRU (Fluegel, 1995). The HRU delineation for the Sandspruit catchment can be seen in Figure 2.2. The catchment is divided into 1 660 HRUs, each represented by a different coloured block in the figure.

The JAMS/J2000-NaCl model includes a variety of land use and management practices which enables the effects of various *what-if* scenarios on the catchment hydrosalinity balance to be simulated (Bugan *et al.*, 2012b). In other words, it allows for the land use of each HRU to be altered (Bugan *et al.*, 2013). There are 100 land use options included in the JAMS/J2000-NaCl hydrological model,

some of which have the potential to be grown in the Sandspruit catchment.

2.3 The current situation

At present, changes to land use are manually inserted into the hydrological model, for each HRU. Researchers adopt a scenario analysis approach to evaluate the impacts of alternative vegetation types as well as the spacial distributions of these, on the water catchment. The three re-vegetation scenarios to be evaluated are according to: riparian zones, contour banks and areas which exhibit a high salt storage in the regolith zone (Bugan *et al.*, 2013). For each scenario, the researcher selects the applicable HRUs and assigns a land use to them. Once a scenario has been entered, the JAMS/J2000-NaCl hydrological model is run to obtain a solution. The researcher then assigns a different land use for that scenario and runs the hydrological model again. Four land uses were evaluated for each scenario. Once all the scenarios have been performed, the results can be compared.

Figure 2.3 is associated with Scenario 1. It shows the HRUs which represent the riparian zone in the Sandspruit catchment. As can be seen in the figure, a riparian zone is the area located along a river, in this case the Sandspruit River. Figure 2.4 is associated with Scenario 2. It shows the HRUs which contain contour banks in the Sandspruit catchment. As can be seen in the figure, contour banks are plentiful in the catchment. They are constructed to prevent soil erosion. Figure 2.5 is associated with Scenario 3. It shows the HRUs which exhibit high (mean $>$ 100 t ha^{-1}) regolith salt storage in the Sandspruit catchment (Bugan *et al.*, 2013).

2.4 Problem description and value to be added

In order to improve the current method of assigning land use in the Sandspruit catchment, the drawbacks of the scenario approach need to be identified. The first shortcoming of the scenario method is that it is time consuming to execute because a land use needs to be individually assigned to each of the 1 660 HRUs, according to the current scenario. This needs to be repeated for different land

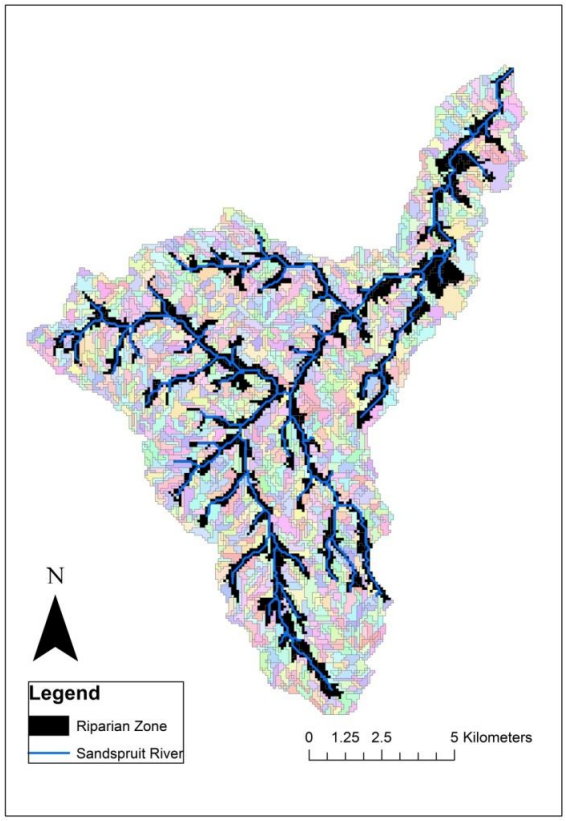


Figure 2.3: Scenario 1: Riparian zones.

uses and each scenario. The second drawback of the scenario approach is that it may not produce the best possible results (set of land uses) as it only tests a small portion of the decision space. It only tests 12 combinations (3 scenarios $\times$ 4 crop types) of land uses for the catchment.

This study aims to replace the scenario analysis approach with an optimisation model that remotely triggers the JAMS/J2000-NaCl hydrological model. The CEM will be used to evaluate proposed solutions obtained from the output of the hydrological model, based on a fitness evaluation. In doing so, an optimal land use configuration for the catchment area is determined, to minimise the salt discharge in the Sandspruit catchment.

This leads to identifying the value propositions of the research. This optimisation model can save the water manager's time, be used repeatedly and expanded

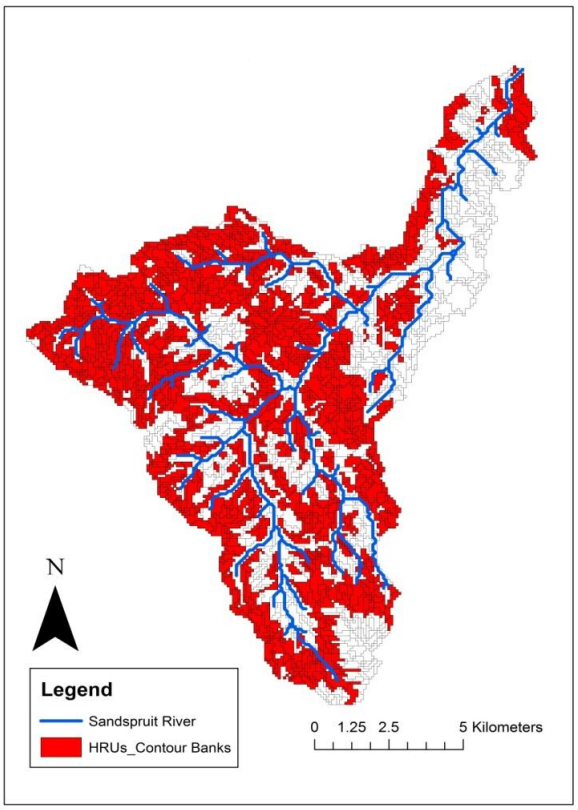


Figure 2.4: Scenario 2: Contour banks.

easily. Furthermore, it will test a far greater decision space than the manual scenarios. This is discussed further in the following section. By testing more combinations of land uses, guidelines for regulating land use can more accurately be developed, in an attempt to reduce the mobilisation of salts to the Berg River.

2.5 Formulation of the optimisation model

In this section, the optimisation model is formed by integrating the JAMS/J2000-NaCl model and the optimisation algorithm. The optimisation model generates input variables for the JAMS/J2000-NaCl model which acts as a black-box. The output obtained from the JAMS/J2000-NaCl model is used to determine performance and subsequently by the CEM to update the input variables. This process

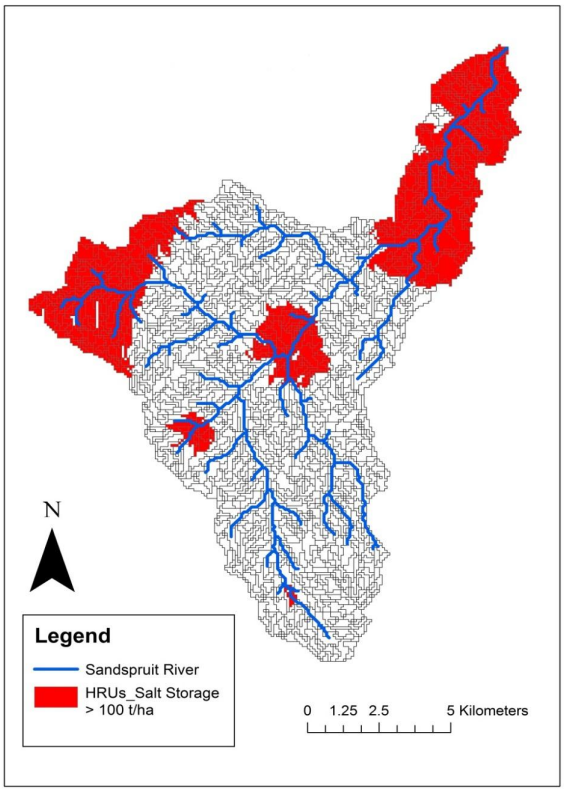


Figure 2.5: Scenario 3: High salt storage in the regolith zone.

continues until a termination condition is reached.

2.5.1 JAMS/J2000-NaCl structure

Taking into account all of the available crop types for the JAMS/J2000-NaCl model and some knowledge of vegetation and farming practices, the crops that will be predominately used in the HRUs can be predicted. Requirements of the ideal vegetation are that they are perennial and have a deep root system (Bugan *et al.*, 2013). The potential crops for the Sandspruit catchment can be found in Table 2.1 with their corresponding crop identifications (CIDs).

For the Sandspruit catchment, all possible crop rotations from these crops were constructed, in conjunction with a subject matter expert. A crop rotation typically consists of either a continuous crop or a three-year planting rotation

Table 2.1: Crop identifications.

CID	Crop name
8	Forest – Evergreen
12	Pasture
14	Winter Pasture
15	Range Grasses
16	Range Brush
28	Winter Wheat
88	Winter Rape

where cultivation takes place every third year and the land is left fallow for the two years in-between. The land is left fallow to restore soil fertility and is often used for grazing (Bugan *et al.*, 2012a). A continuous crop is the same crop type in rotation indefinitely. Land uses that have continuous crops are trees (Forest - Evergreen) and the natural vegetation — Renosterveld (Range Brush), as they are more permanent. The crop rotations suitable for the Sandspruit catchment are shown in Table 2.2 in terms of their CIDs. Each crop rotation is associated with a rotation identification (RID). As can be seen in the table, one year represents one crop type for the period 2006 to 2010. An RID is assigned to each HRU in the catchment to model the land uses that should be implemented in that HRU for a number of years.

2.5.2 Optimisation model using the cross-entropy method

The land use for each HRU becomes a decision variable in the optimisation model and the possible decision variable values are the RIDs. A combination of these RIDs results in a yield of salt for the catchment.

The water management problem can be classified as a deterministic combinatorial optimisation problem. In this type of problem, the decision maker seeks the appropriate combination of values that optimises the objective function. In combinatorial optimisation, the decision space rapidly (exponentially) increases as the solution space increases. This problem has $13^{1\,660}$ possible combinations of crop rotations for the catchment.

Table 2.2: Crop rotations.

RID	2006	2007	2008	2009	2010
2	28	28	28	12	12
3	28	28	28	14	14
4	28	28	28	28	28
5	16	16	16	16	16
6	8	8	8	8	8
7	28	12	88	12	28
8	28	12	28	12	28
9	28	12	12	12	12
10	8	28	28	28	28
11	16	28	28	28	28
12	28	28	28	15	15
13	28	28	28	88	28
14	28	88	28	88	28

Although the JAMS/J2000-NaCl model provides its users with a range of output values, this study is only concerned with a single output variable — the salt discharge from the catchment for the period analysed. The objective of the optimisation model is to minimise this salt output.

The optimisation model was implemented in Matlab® (primary program) and constructed to interact with the JAMS/J2000-NaCl model (secondary program) to update the input variables, receive output and run the hydrological simulation for each iteration. The JAMS/J2000-NaCl model was executed by remotely triggering the batch script for JAMS with an application programming interface (API).

For each iteration, the JAMS/J2000-NaCl model was run for the period 01/01/2006 to 31/12/2010. The period 01/01/2006 to 31/12/2008 is used as an initialization period where there are start-up conditions and uncertainty in the model. The period 01/01/2009 to 31/12/2010 is the calibration period (de Clercq *et al.*, 2013). For this study, the model output was only evaluated for the calibration period as this is when the researchers start considering the results.

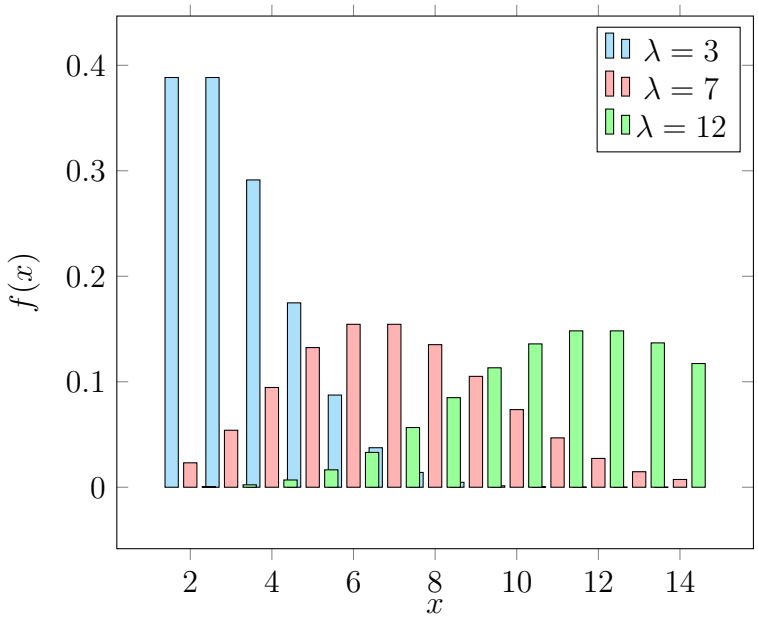


Figure 2.6: Truncated Poisson distribution on $2 \leq x \leq 14$.

The truncated Poisson distribution was used as the sampling distribution for the CEM. It is truncated for the region of the decision variable range. Examples of truncated Poisson distributions with different λ parameters in the range, are shown in Figure 2.6.

The optimisation model runs the JAMS/J2000-NaCl hydrological model with a population size of 20 for 20 generations, resulting in a total of 400 evaluations. As a result of the JAMS/J2000-NaCl model having to be rerun many times, a disadvantage of the optimisation model is its extremely lengthy operating time. The pseudo-code for the optimisation model is shown in Algorithm 2.

2.6 Verification of the optimisation model

In order to verify that the CEM was correctly applied, the algorithm coded was tested with different deterministic objective functions. In all cases, the optimal decision variables were found.

The optimisation model was initially built using a modular programming technique. Each of the three modules developed were separately debugged to ensure

Algorithm 2 Land use optimisation with the CEM

Input: Let R be the number of replications of the simulation, N the number of evaluations in each replication and $[l_l, l_u]$ the range of RIDs, $\alpha = 0.3$ and the percentage of samples to include in the elite $\varrho = 20\%$.
Generate an initial vector $\boldsymbol{\lambda}$, within the RID range.
for all R **do**
    for $i = 1 \rightarrow N$ **do**
        for $j = 1 \rightarrow 1660$ **do**
            Draw $u = U[0, 1]$
            $F = 0$
            for $x = l_l \rightarrow l_u$ **do**
                $f = (e^{-\lambda}\lambda^x)/x! \left(\sum_{x=l_l}^{x=l_u} \frac{e^{-\lambda}\lambda^x}{x!}\right)$
                $F = F + f$
                if $u \leq$ F **then**
                    return RID$(i, j) = x$.
                end if
            end for
        end for
    end for
    for $n = 1 \rightarrow N$ **do**
        Run JAMS/J2000-NaCl for the RIDs in row n.
        Store the salt output in RID$(n, 1661)$.
    end for
    Rank the population in ascending order, according to the salt outputs.
    Calculate the mean of the elite vector for each HRU using ϱ.
    Update $\boldsymbol{\lambda}$ using the smoothing function.
end for

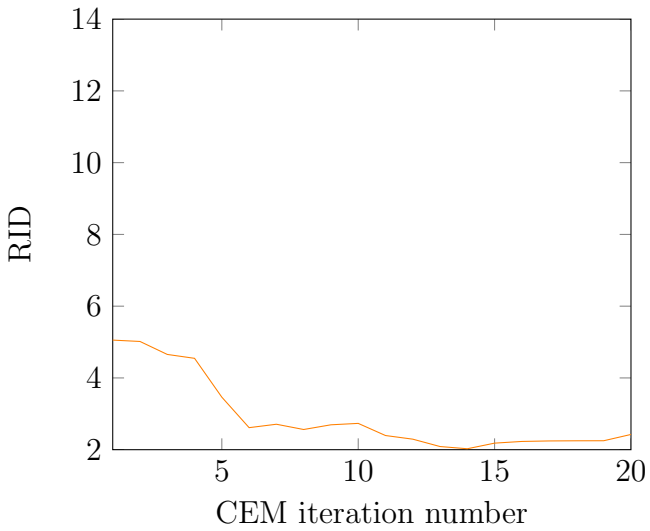


Figure 2.7: Progression of λ for HRU 732.

that they worked correctly. All additional compiler and run time errors were corrected during program execution. Many tests were performed to ensure that the coded Matlab® program communicated with the JAMS/J2000-NaCl model correctly and that the input data updated accordingly.

2.7 Experimental results of the water management problem

This section provides a summary of the results obtained from experimentation. As previously explained, the CEM algorithm updates parameter vector $\boldsymbol{\lambda}$ to an improved solution after each evaluation of the simulation model.

For each of the 1 660 HRUs there exists a unique graph, such as the one in Figure 2.7. The author chose four examples to present (Figures 2.7 to 2.10).

First, the progression of λ for HRU 732 is illustrated in Figure 2.7. It shows how the λ parameter of the Poisson distribution initially fluctuates then stabilises as the simulation progresses. For each HRU, the value of λ should converge, and the value at the final iteration is taken as the RID solution. For example, HRU 732 in Figure 2.7 is assigned RID 2 at the end of the simulation run.

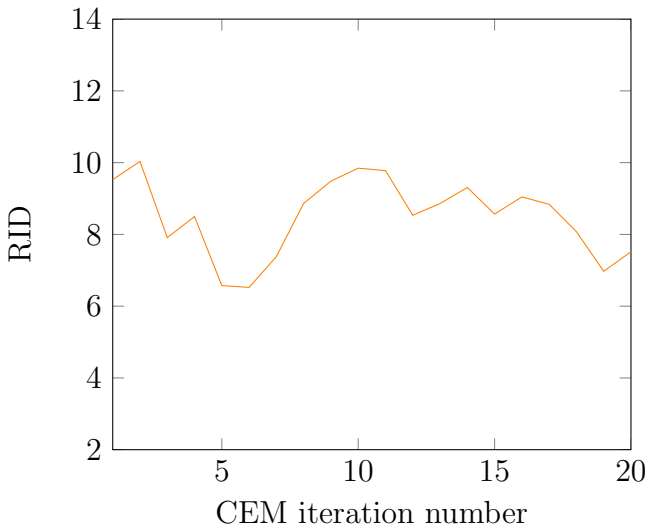


Figure 2.8: Progression of λ for HRU 858.

Figure 2.8 shows how the value of λ varies for HRU 858. When comparing Figure 2.7 to Figure 2.8, it can be seen that not all the λ parameters converge as well as for HRU 732. The final crop rotation that was assigned to HRU 858 is RID 8. Figure 2.9 and Figure 2.10 are further examples of how the algorithm allocates crop rotations. They show that HRU 1 639 is assigned RID 3 and HRU 337 is assigned RID 8.

Figure 2.11 shows the average of the four smallest values (quantile of the population) of the salt output, for each generation of the simulation. The salt output is obtained from the hydrological model of the catchment. As can be seen in the figure, the salt output decreases as the evaluations elapse. This makes sense as the aim of the optimisation model is to decrease the salt output.

The frequency of each crop rotation assigned throughout the catchment, is shown in Figure 2.12. As can be seen in the figure, RID 2 and RID 3 should be predominantly cultivated in the catchment area.

From the convergence of $\boldsymbol{\lambda}$, the near-optimal solution of crop rotations for each HRU has been identified. Putting the combination of these crop rotations for the catchment into the JAMS/J2000-NaCl model, a final salt output of 28 267 tons is achieved. Data collected at the catchment for the same years (2009 and 2010) produces a salt output of 29 385 tons (Bugan *et al.*, 2013). According to

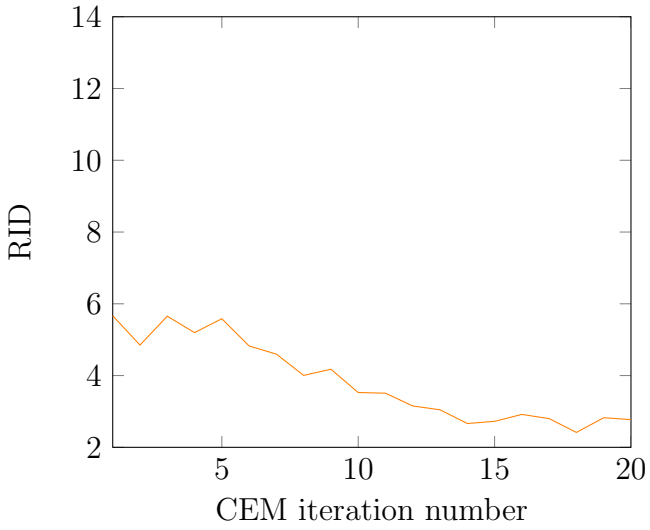


Figure 2.9: Progression of λ for HRU 1 639.

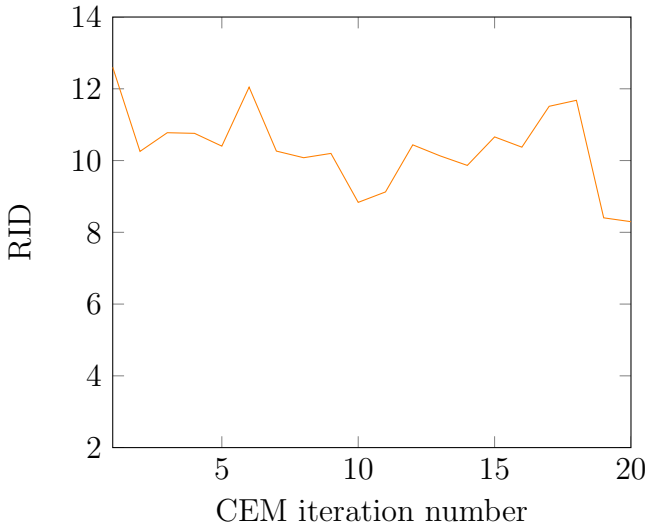


Figure 2.10: Progression of λ for HRU 337.

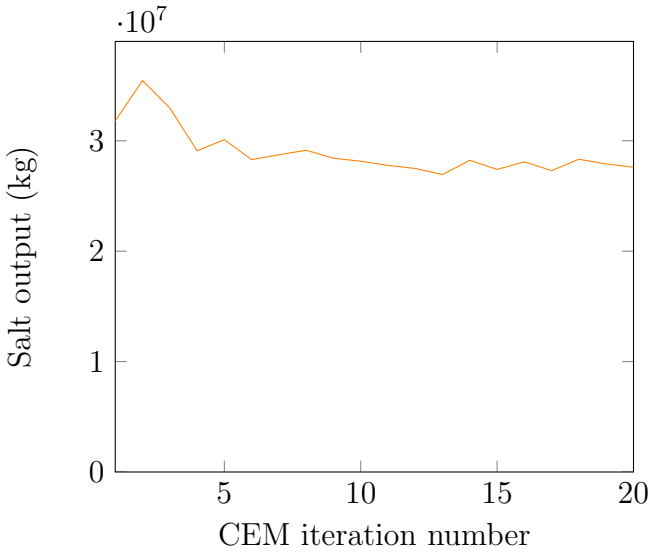


Figure 2.11: Progression of the objective function for the average of the four smallest values.

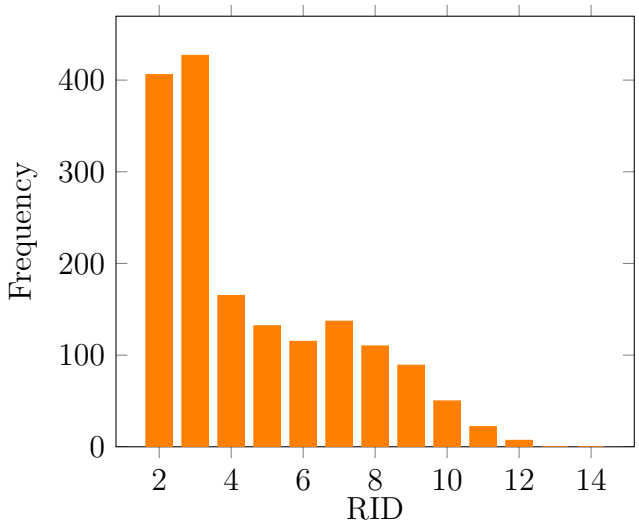


Figure 2.12: The total number of each RID assigned to HRUs.

a hydrologist at the Council for Scientific and Industrial Research, this results in a 3.8% reduction in the catchment salt output which is a small improvement (Bugan, telephonic consultation, 28/08/2013).

From experimentation it was found that the optimisation model performed well. Further investigation should aim to enhance the integration of JAMS/J2000-NaCl and the CEM for optimisation, as it is valuable water management research.

2.8 Future work relating to the water management problem

There are five improvements to the optimisation model discussed in this section. They can act as the way forward for research outputs on this topic.

1. The simulation should be repeated with the full set of crop types that are suited to the Sandspruit catchment. This list of crops should at least include:
   - rye,
   - oats,
   - winter rye, and
   - alfalfa (lupins).
2. An investigation could take place to determine if there is any connection between certain crop rotations and the areas to which they were assigned. For example the algorithm could be assigning specific crops to riparian zones, contour banks or areas which exhibit a high salt storage in the regolith zone.
3. To give the model more credibility, further work can involve the farmers of the area to ensure that they have confidence in the model's results.
4. The optimisation model currently takes a long time to complete execution. The next step in this project could be to coordinate with the JAMS/J2000

developers at the Friedrich-Schiller-University of Jena (Germany) to attempt to reduce the hydrological model execution time. This will assist in making the optimisation model more feasible in practice.

5. For the CEM, the hydrological model was run for 20 evaluations and 20 generations. More iterations could be performed to determine if the solution will improve.

2.9 Conclusion: Chapter 2

In this chapter, the CEM was applied to a practical water management problem. By doing so, the fundamentals of the CEM were studied, which will be of assistance later in the study when the multi-objective optimisation with the CEM is extended and applied.

From the water management problem, it is concluded that combining an optimisation algorithm with the JAMS/J2000-NaCl model can obtain an improved solution, in terms of the salt yield of the catchment. With slight modifications, the optimisation model built can be applied to water catchment areas around the world that have already been modelled by the JAMS/J2000 model, and used as a guideline when forming land use regulations.

The next chapter is concerned with reviewing and comparing multi-objective ranking and selection procedures for a suitable procedure for the multi-objective optimisation method with the CEM by (Bekker & Aldrich, 2010).

Chapter 3

Multi-objective ranking and selection

In the previous chapter, the cross-entropy method was applied to a single-objective optimisation problem. This chapter advances to multi-objective optimisation (MOO) and a brief overview on this topic is given.

This chapter focuses on optimisation of relatively small problems where there are a known number of alternatives that are to be ranked and the best alternative selected. As the search space is small, there is no need for search algorithms and they are only incorporated in Chapter 4.

An introduction is presented on single-objective ranking and selection, where the topic is divided into three classes. Some methods are explained for each class.

Three multi-objective ranking and selection methods are found in literature, that are compatible with the aim of this research. These methods are presented in detail, as this is the core of the study. The methods are the *multi-objective optimal computing budget allocation*, the *multi-objective optimal computing budget allocation with an indifference-zone* and the *two-stage-Pareto-set-selection procedure*. The first two algorithms are then tested on two problems, which are described in the experimental design. How to measure the performance of the experiments is discussed, followed by the parameter settings of the algorithms.

Lastly the results and assessment of the multi-objective ranking and selection algorithms is presented.

3.1 Introduction to multi-objective optimisation

Every day decisions commonly result in several simultaneous outcomes. Such a decision can be said to have multiple performance measures, which are often conflicting and non-commensurate. A brief introduction to MOO is presented in this section. The mathematical definitions and concepts presented are based on Coello Coello (2009).

Multi-objective problems (MOPs) have two or more conflicting objectives that are required to be optimised simultaneously, while satisfying a specified set of constraints. The process of solving the MOP is known as MOO. A MOP has a set or a vector of solutions, where each solution is a trade-off between the objectives. In 1896, Vilfredo Pareto formally defined this set of solutions as the Pareto optimum. The solution to a MOP is defined as Pareto optimal if there exists no other feasible solution that would be better in some criterion, without simultaneously causing at least one other criterion to be worse (Coello Coello *et al.*, 2007). To put it simply, there are no solutions that are better for the certain input variables and constraints. Pareto dominance is the term used to define one set of solutions as being better than another (Goldberg, 1989). The solutions in the Pareto optimal set are non-dominated as none of the points dominate each other.

The canonical formulation of the MOO problem with H objectives and $M+Q$ constraints is:

$$\begin{aligned} \text{Minimise } \mathbf{f}(\mathbf{x}) &= [f_1(\mathbf{x}), f_2(\mathbf{x}), \ldots, f_H(\mathbf{x})]^T && (3.1)\\ \text{subject to } \mathbf{x} &\in \Omega && (3.2)\\ \Omega &= \{\mathbf{x} \mid g_i(\mathbf{x}) \leq 0,\ i = 1, 2, \ldots, M; && (3.3)\\ h_j(\mathbf{x}) &= 0,\ j = 1, \ldots, Q\}. && (3.4) \end{aligned}$$

where $\mathbf{x} = [x_1, x_2, \ldots, x_D]^T$ is a D dimensional vector of decision variables for which numerical quantities are to be chosen in the optimisation problem.

The following definitions pertaining to Pareto optimality are defined (Coello Coello, 2009):

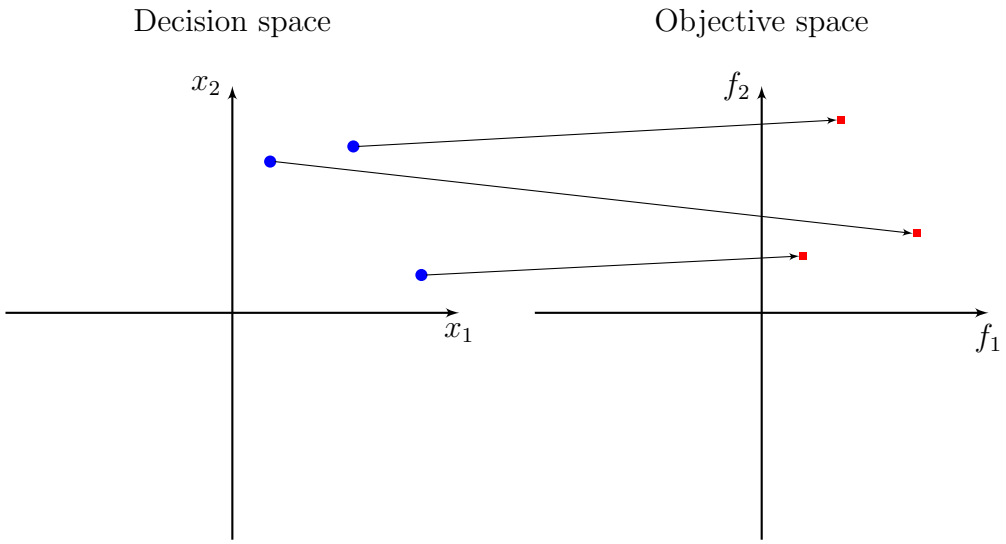


Figure 3.1: Two Euclidian spaces for multi-objective optimisation.

Definition 1: Given two vectors $\mathbf{u} = (u_1, \ldots, u_H)$ and $\mathbf{v} = (v_1, \ldots, v_H) \in \mathbb{R}^H$, then $\mathbf{u} \leq \mathbf{v}$ if $u_i \leq v_i$ for $i = 1, 2, \ldots, H$, and $\mathbf{u} < \mathbf{v}$ if $\mathbf{u} \leq \mathbf{v}$ and $\mathbf{u} \neq \mathbf{v}$.

Definition 2: Given two vectors $\mathbf{u}$ and $\mathbf{v}$ in $\mathbb{R}^H$, then $\mathbf{u}$ dominates $\mathbf{v}$ (denoted by $\mathbf{u} \prec \mathbf{v}$) if $\mathbf{u} < \mathbf{v}$.

Definition 3: A vector of decision variables $\mathbf{x}^* \in \Omega$ (Ω is the feasible region) is *Pareto optimal* if there does not exist another $\mathbf{x} \in \Omega$ such that $\mathbf{f}(\mathbf{x}) \prec \mathbf{f}(\mathbf{x}^*)$.

Definition 4: The *Pareto optimal set* $\mathcal{P}^*$ is defined by $\mathcal{P}^* = \{\mathbf{x} \in \Omega \mid \mathbf{x} = \mathbf{x}^*\}$.

Definition 5: The *Pareto front* $\mathcal{P}_T^*$ is defined by $\mathcal{P}_T^* = \{\mathbf{f}(\mathbf{x}) \in \mathbb{R}^H \mid \mathbf{x} \in \mathcal{P}^*\}$. The vectors in $\mathcal{P}^*$ are called *nondominated*, and there is no $\mathbf{x} \in \Omega$ such that $f(\mathbf{x})$ dominates $f(\mathbf{x}^*)$.

To solve the MOO problem, the Pareto optimal set is found, by searching through all the decision variable vectors that satisfy the constraints, for the set that optimises the objective function vector. According to the constraints of the problem, the range of possible decision variables can be determined.

Figure 3.1 displays the decision space consisting of two decision variables, x_1 and x_2, and the objective space consisting of two objectives, f_1 and f_2. This is an example of an unconstrained problem. As can be seen in the figure, each vector in the decision space is associated with a vector in the objective space. A

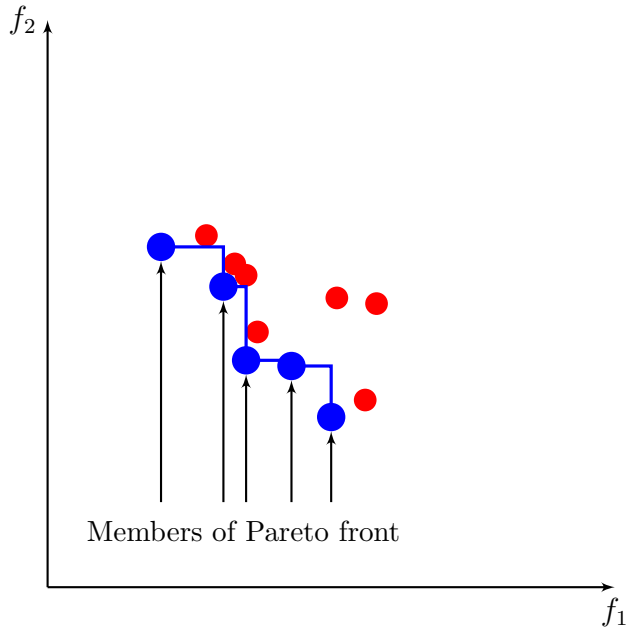


Figure 3.2: Pareto front explained for two minimised objectives.

combination of the decision variable values is evaluated to obtain a realisation for the objective function. The Pareto set consisting of all the non-dominated solutions is determined from these points in the objective space. This set can be shown graphically as the Pareto front. An example of a Pareto front (blue dots) is shown in Figure 3.2, where both objectives are to be minimised.

To determine the Pareto optimal set, the solutions have to be ranked to distinguish the good solutions from the bad ones. The most popular ranking method in literature is the Pareto ranking, established in Goldberg (1989). The algorithm is presented next because it plays an important role in this study.

The following working matrix $\mathbf{W}$ is provided for the algorithm (shown in Table 3.1). The working matrix consists of N rows and $D + H + 1$ columns, where N is the number of scenarios to rank, D is the number of decision variables and H is the number of objectives. Observations of the first decision variable are stored in column 1, the second in column 2, and so on until column D. The objective function values are stored in columns $D + 1$ to $D + H$ and the rank of each solution is stored in the last column. The pseudo-code for the ranking process is presented in Algorithm 3.

Table 3.1: Structure of the working matrix

Decision variables				Objectives				Rank
X_{11}	X_{12}	$\dots$	X_{1n}	f_{11}	f_{12}	$\dots$	f_{1m}	ρ_1
$\vdots$	$\vdots$		$\vdots$	$\vdots$	$\vdots$		$\vdots$	$\vdots$
X_{S1}	X_{S2}	$\dots$	X_{Sn}	f_{S1}	f_{S2}	$\dots$	f_{Sm}	ρ_S

Algorithm 3 Pareto ranking algorithm (Minimisation)

1: Input: Working matrix $\mathbf{W}$ with N rows and $D + H + 1$ columns, and user-selected threshold ρ_E.
2: $j \leftarrow D + 1$.
3: Sort the working matrix $\mathbf{W}$ with the values in column j in *descending* order.
4: $r_p \leftarrow 1$.
5: $r_q \leftarrow r_p$.
6: If $\mathbf{W}(r_p, j+1) \geq \mathbf{W}(r_q+1, j+1)$, increment the rank value ρ_{r_p} in $\mathbf{W}(r_p, D+H+1)$.
7: $r_q \leftarrow r_q + 1$.
8: If $\mathbf{W}(r_p, D+H+1) < \rho_E$ and $r_q < N$, return to Step 6.
9: $r_p \leftarrow r_p + 1$.
10: If $r_p < N$, return to Step 5.
11: $j \leftarrow j + 1$.
12: If $j < D+H-1$, return to Step 3, otherwise return the rows in $\mathbf{W}$ with rank value not exceeding ρ_E as the non-dominated vector *Elite*.

A ranking value of a solution indicates the number of other solutions in the population that dominate it. A solution that possesses a ranking value of zero is a non-dominated solution, as no other solution dominates it. After ranking all the solutions, the method assigns solutions with a ranking value less than a threshold value ρ_E to the Pareto set.

Approaches to solving MOPs include: the weighted sum approach, multiattribute utility analysis, lexicograohic approaches, multi-objective metaheuristics and goal programming (De Weck, 2004). The focus of this study is on multi-objective metaheuristics.

3.2 Introduction to ranking and selection

Ranking and selection (R&S) procedures aim to identify the desired scenario, where the desired scenario could either be the single best scenario amongst a set of alternatives, the feasible scenarios under constraints, or the best feasible subset. The single-objective R&S overview in the following section is limited to the selection of the best scenario. This study excludes comparing all alternatives against a standard. A standard could be the existing system or operating policy in place.

3.2.1 Single-objective ranking and selection

Simulation optimisation where the solution space is relatively small and finite is known as R&S. It is assumed that a finite number of feasible alternatives are given prior to the application of an R&S procedure (Juxin, 2012). R&S procedures are statistical methods that select the best scenario from a set of competing alternatives. They do this by determining the number of simulation replications required to guarantee a certain measure of selection quality, at a pre-specified level, with the least possible computational expense. Popular measures of selection quality include the probability of correct selection, the expected opportunity cost and the expected net present value (Lee *et al.*, 2010b). The most recent reviews of R&S found in literature are Branke *et al.* (2007), Kim & Nelson (2007) and Lee *et al.* (2010a).

Four solution approaches to these types of problems are (Swisher *et al.*, 2003):

- the optimal computing budget allocation framework,
- indifference-zone methods,
- decision theoretic methods, and
- subset selection.

The approaches differ in how replications are allocated to certain scenarios.

Decision theoretic methods select the additional number of simulation replications that will maximise the expected value gained from those replications (Chick,

1997; Chick *et al.*, 2001). Decision theoretic methods will not be discussed further because no multi-objective decision theoretic methods exist, and therefore the method is not in-line with the ultimate aim of this study.

Subset selection eliminates non-competitive scenarios and constructs a subset of at most m scenarios that contains the best scenario, where m is a predetermined number of scenarios. Additionally, it contains this best scenario with a pre-specified level of confidence (Gupta, 1965). Subset selection procedures are only discussed as part of an indifference-zone procedure.

Older R&S procedures recommend a solution space of 2 to 20 scenarios, while modern procedures can handle a few hundred scenarios (Kim & Nelson, 2007).

3.2.1.1 Indifference-zone methods

Indifference-zone (IZ) procedures are based on a Frequentist view, which determines the best scenario with a guarantee on the probability of correct selection. An IZ is the smallest change in an objective function value that is significant to the stakeholder. That is, if the change between two scenarios is smaller than the IZ, then the stakeholder does not have a preference between the two scenarios and they are indifferent (Swisher *et al.*, 2003).

IZ R&S procedures have their origin in Bechhofer (1954) from the statistics community where a single stage procedure is proposed, assuming known common variance across all scenarios. As this assumption is usually not valid in practice, Dudewicz & Dalal (1975) established a two-stage procedure for problems with unknown and unequal variances. The procedure takes an initial sample of replications for each scenario and then based on these sample variances, determines the number of additional replications for each scenario at the second stage. Rinott (1978) improves this two-stage procedure to obtain a higher probability of correct selection, which consequently requires more samples.

Due to the requirement on a large number of samples, recent improvements in the field have led to a number of more efficient procedures. The two-stage procedure of Nelson *et al.* (2001) is applied when there is a large number of alternatives. At the first stage, an initial number of replications is performed

for each scenario and subsequently this output is used for subset selection on the scenarios. At the second stage, IZ ranking is applied to the remaining candidates.

Compared to the two-stage procedures discussed, sequential procedures allocate simulation budget across scenarios over more than two stages. The sequential procedure by Kim & Nelson (2001) begins by running an initial number of replications for every scenario. At each stage after this, no more replications are performed for scenarios that are clearly inferior, however, additional replications are performed for scenarios that are still candidates for the best. This process is repeated until the procedure is confident that a particular scenario is the best, or within the specified IZ of the best. What makes the method superior is that at an early stage it eliminates some competing scenarios, thereby greatly reducing the overall computational effort.

Alrefaei & Alawneh (2004) developed a two-stage procedure that screens scenarios in the first stage, to obtain a set that has a high probability of being the best. In the second stage, an IZ ranking procedure is applied to the remaining candidates.

Chen & Kelton (2005) aim to avoid extensively relying on sample mean estimates from only one stage (the first stage in a two-stage procedure). At each iteration, their sequential procedure dynamically computes the incremental sample size of scenarios based on the difference between sample means, sample variances, and the IZ at that iteration.

As most of the procedures initially perform a number of replications for all scenarios, then repeatedly obtain small incremental samples from the scenarios, simulation software that is capable of this is ideal.

3.2.1.2 The optimal computing budget allocation framework

Another line of R&S methods exist, known as optimal computing budget allocation (OCBA) algorithms (Chen *et al.*, 1996, 2000a, 2003, 1997, 2000b).

OCBA procedures follow a Bayesian methodology and attempt to allocate a finite computing budget so as to maximize the probability of correct selection. The procedures only simulate likely competitors for the best scenario leading to computational savings. To identify the candidates for the best and the number

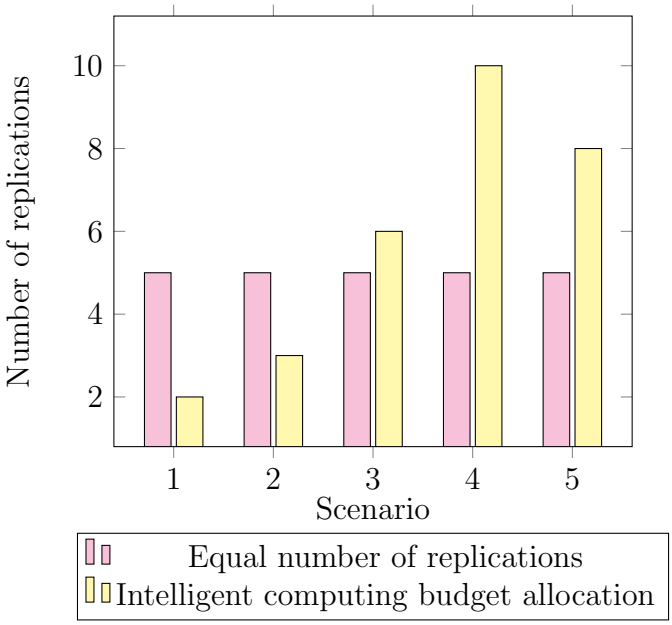


Figure 3.3: Comparison of simulation budget allocations for obtaining the same confidence level (Chen *et al.*, 1996).

of replications to allocate to them, the procedures makes use of both sample mean and sample variance information, and derive allocation rules under certain asymptotic conditions. Figure 3.3 shows an example of a computing budget that was allocated using an OCBA procedure, compared to an equal allocation of replications.

For IZ procedures, the default stopping rule is to continue sampling until the specified measure of selection quality has been reached. The default stopping rule for OCBA procedures is to continue sampling until the given computing budget is exhausted. The IZ rule can be more efficient because it can terminate if the evidence of correct selection has reached its target, rather than only when the computing budget is exhausted. For OCBA procedures, the total computing budget might be tricky to choose and the experimenter will not know what selection quality will be returned at a certain simulation budget. Whereas the selection quality parameter in IZ procedures is easier for experimenters to understand.

Chen *et al.* (2004) combined the IZ concept and the OCBA framework. The allocation for the procedure is calculated differently to the standard OCBA rules

by replacing the difference in sample means of two scenarios by the given IZ value, if the change is smaller than the IZ value.

3.2.2 Multi-objective ranking and selection

In most practical problems, more than one performance measure of a system is considered. In this case, the problem becomes a multi-objective ranking and selection (MORS) problem.

Some solution approaches to the MORS problem exist that transform the problem into a single-objective case. Having done this, one of the single-objective R&S methods is applied to the problem (Lee *et al.*, 2006). Morrice *et al.* (1998) and Butler *et al.* (2001) combine *multi-attribute utility theory* (MAUT) with the IZ approach by Rinott (1978) to form an R&S procedure to select the best scenario from a set of possible scenarios, for multiple performance measures. Decision makers are required to specify weights for each performance measure which are used in aggregating the utility function. This transforms the multiple performance measures into a single-objective. The procedure then finds the scenario with the highest utility — this is the best scenario. This procedure yields a single best solution, however in the MORS problem the optimal solution is the Pareto optimal set. In this case, to determine the set of non-dominated solutions, we seek procedures that solve the MOP directly.

Mattila & Virtanen (2014) develop two MORS methods using incomplete preference information. In the methods, the performance measures are aggregated with a multi-attribute utility function which takes the incomplete preference information into account to determine the preferentially non-dominated scenarios. The preference information is incomplete in the sense that weights, reflecting importance of objectives, are specified as intervals, instead of exact values.

The first procedure, OCBA-a, selects non-dominated scenarios using an absolute dominance relationship. The existing OCBA procedure is used to maximise the probability of selecting the absolute dominated scenarios, limited to a simulation budget. The second procedure, MOCBA-p, identifies non-dominated scenarios according to pairwise dominance. Scenario a dominates scenario b according to pairwise dominance if the expected utility of scenario a is higher than

the expected utility of scenario b over all feasible weights (Weber, 1987). The existing multi-objective optimal computing budget allocation (MOCBA) procedure (discussed in the next section) is applied to maximise the probability of correctly selecting the set of pairwise non-dominated scenarios with a limited computing budget. Both methods are shown to be more accurate in determining their respective non-dominated scenarios and lead to significant computational savings, when compared to another MOCBA procedure. MOCBA-p also obtains a smaller number of non-dominated scenarios which makes selection easier for the decision maker. The advantage of OCBA-a is its straightforward approach and ease of implementation.

In the methods by Mattila & Virtanen (2014), the non-dominated solutions identified would be strongly dependent on the preferences of the decision maker. If another decision maker has different preferences, or the preferences of the decision maker were to change, the solutions may become inferior. This is deemed a limitation. As a result, this study only considers a posteriori methods, where user preferences are only taken into account after the results of an optimisation model is known. Three methods have been identified in literature for MORS problems to obtain a Pareto optimal set with no preference information. These methods are the MOCBA algorithm (Lee *et al.*, 2004, 2010b), the MOCBA with IZ (MOCBA_IZ) framework (Teng *et al.*, 2010) and the *two-stage-Pareto-set-selection* (TSPS) procedure (Chen & Lee, 2009). These methods are described in detail in the following section.

3.3 The MORS algorithms under investigation

The literature on the MORS algorithms considered in this study is discussed next. Assume, without loss of generality, the aim of the optimisation in this section is to minimise all the objectives.

3.3.1 Multi-objective optimal computing budget allocation algorithm

The MOCBA algorithm is used to find the Pareto set for multi-objective simulation models. It does this by efficiently allocating simulation replications to the possible scenarios so that the non-dominated set can be found with high confidence and the minimum number of simulation replications (Chen & Lee, 2010). The way MOCBA allocates replications is similar to the OCBA procedure for the single-objective case, apart from the concept of Pareto optimality being incorporated into the OCBA scheme. MOCBA was first introduced in Lee *et al.* (2004) where the number of non-dominated scenarios (called system designs) is known in advance. Usually this is unknown before implementing the solution procedure. This research was later adapted in Lee *et al.* (2010b) and Chen & Lee (2010) to remove this limitation and thus be more realistic. Only the latter versions of MOCBA are discussed in this study. For the proof and derivations of the equations used in the procedure the reader is referred to Lee *et al.* (2010b) and Chen & Lee (2010). Lee *et al.* (2010b) and Chen & Lee (2010) differ in how they formulate the MORS problem, leading to slightly different allocation rules and procedures. There is also an account of the MOCBA algorithm in Lee *et al.* (2006). The procedure varies substantially from the one in Lee *et al.* (2010b). The method was studied and seemed simpler than the MOCBA procedure in Lee *et al.* (2010b), hence it was applied to two models. The author could not obtain good solutions from the algorithm so the research continued with the MOCBA algorithm in Lee *et al.* (2010b).

The problem for MOCBA can be stated as follows. Suppose one has scenarios $i \in \mathcal{S} = \{1, \ldots, N\}$, each estimated by independent objectives $k \in \mathcal{H} = \{1, \ldots, H\}$. Each performance measure is represented by a random variable following a normal distribution. The distribution has a true mean μ_{ik} and known (and finite) variance σ_{ik}^2, for the k-th objective of scenario i. The aim of the procedure is to find the Pareto optimal set of scenarios by running simulations to estimate the true mean of performance measures. Given that R_i simulation observations $(X_{ik}^1, X_{ik}^2, \ldots X_{ik}^{R_i})$ are taken for the k-th objective of scenario i, the sample mean performance is $\bar{X}_{ik} = \frac{\sum_{a=1}^{R_i} X_{ik}^a}{R_i}$. The probability distribution of $\bar{X}_{ik}$

(the sampling distribution of μ_{ik}) is

$$\bar{X}_{ik} \sim N\left(\mu_{ik}, \frac{\sigma_{ik}^2}{R_i}\right). \tag{3.5}$$

When R_i and R_j replications are allocated for scenario i and scenario j then its probability distribution is

$$\bar{X}_{jk} - \bar{X}_{ik} = N\left(\delta_{ijk}, \sigma_{ijk}^2\right), \tag{3.6}$$

where $\delta_{ijk} = \mu_{jk} - \mu_{ik}$ and $\sigma_{ijk}^2 = \left(\frac{\sigma_{ik}^2}{R_i}\right) + \left(\frac{\sigma_{jk}^2}{R_j}\right)$.

3.3.1.1 Dominance relationship and its probability description based on observed performance

The approximated Pareto set is constructed by comparing the performance measures of scenarios and selecting the non-dominated scenarios to form the approximated Pareto set. Scenario j dominates scenario i, denoted by $j \hat{\prec} i$ if the following condition holds

$$\begin{aligned} &\text{for every objective } k \in \mathcal{H}, \bar{X}_{jk} \leq \bar{X}_{ik} \\ &\text{and for at least one } k \in \mathcal{H}, \bar{X}_{jk} < \bar{X}_{ik}. \end{aligned} \tag{3.7}$$

Scenarios i are assigned to the approximated Pareto set such that for $j \hat{\prec} i$, a scenario j cannot be found. This is implemented using the Pareto ranking algorithm based on work by Goldberg (1989). The probability that scenario j dominates scenario i is calculated by

$$P(j \hat{\prec} i) = \prod_{k=1}^{H} P(\bar{X}_{jk} \leq \bar{X}_{ik}) = \prod_{k=1}^{H} \Phi\left(-\frac{\delta_{ijk}}{\sigma_{ijk}}\right), \tag{3.8}$$

where Φ is the standard cumulative normal distribution.

A performance index is defined to measure how non-dominated scenario i is and is described by,

$$\psi_i = P\left(\bigcap_{\substack{j \in \mathcal{S} \\ j \neq i}} (j \hat{\not\prec} i)\right), \tag{3.9}$$

where $(j \hat{\not\prec} i)$ denotes that scenario j does not dominate scenario i.

The performance index ψ_i therefore measures the probability that i is not dominated by any of the other scenarios. If ψ_i is near one then the probability that other scenarios are better than scenario i is low and the probability that other scenarios are worse than scenario i is high, because they are being dominated. Therefore when $\psi_i = 1$, scenario i is non-dominated. As ψ_i cannot be explicitly expressed, it can be bounded by

$$\prod_{\substack{j\in\mathcal{S}\\ j\neq i}}\left[1-\prod_{k=1}^{H}P(\bar{X}_{jk}\leq\bar{X}_{ik})\right]\leq\psi_i\leq\min_{\substack{j\in\mathcal{S}\\ j\neq i}}\left[1-\prod_{k=1}^{H}P(\bar{X}_{jk}\leq\bar{X}_{ik})\right]\quad\text{for } i\in\mathcal{S}. \tag{3.10}$$

3.3.1.2 Two types of errors of the approximated Pareto set

Due to the estimated performance of scenarios from stochastic simulation, it cannot be known for sure that scenarios in the approximated Pareto set (S_p) are actually non-dominated (and scenarios in the approximated non-Pareto set ($\bar{S}_p$) are dominated). To represent this uncertainty, a Type I and Type II error is set up to evaluate the quality of the approximated Pareto set. The concept is taken from the field of statistics. Type I error e_1 occurs when at least one scenario in $\bar{S}_p$ is classified as non-dominated. It relates to the probability that at least one observed dominated scenario is actually non-dominated. Let E_i be the event that scenario i is non-dominated by all other scenarios and E_i^c be the complement. By (3.9) it is known that $P(E_i) = \psi_i$ and $P(E_i^c) = 1 - \psi_i$. The probability that every scenario in $\bar{S}_p$ is dominated by at least one other scenario is given by

$$P\left(\bigcap_{i\in\bar{S}_p}E_i^c\right). \tag{3.11}$$

Therefore, the probability of Type I error is

$$e_1 = 1 - P\left(\bigcap_{i\in\bar{S}_p}E_i^c\right). \tag{3.12}$$

Type II error e_2 occurs when at least one scenario in S_p is dominated by another scenario(s). It relates to the probability that at least one non-dominated

scenario is actually dominated by another scenario. The probability that all scenarios in S_p are non-dominated is given by

$$P\left(\bigcap_{i\in S_p} E_i\right). \tag{3.13}$$

Therefore the probability of Type II error is

$$e_2 = 1 - P\left(\bigcap_{i\in S_p} E_i\right). \tag{3.14}$$

Lee *et al.* (2010b) bounds the two types of errors from above by the approximated Type I error ae_1 and approximated Type II error ae_2 as described by

$$e_1 \leq ae_1 = \sum_{i\in \bar{S}_p} \psi_i \tag{3.15}$$

and

$$e_2 \leq ae_2 = \sum_{i\in S_p} (1-\psi_i). \tag{3.16}$$

The error ae_1 can be interpreted as the sum of the probabilities, for each scenario in $\bar{S}_p$, that it is not dominated by any other scenario. It will equal zero when all scenarios in $\bar{S}_p$ are dominated. The error ae_2 can be interpreted as the sum of the probabilities, for all the scenarios in S_p, that scenario i is dominated by other scenarios. It will equal zero when all scenarios in S_p are non-dominated. Hence, the approximated Pareto optimal set approaches the Pareto optimal set when both the errors approach zero. From (3.10) the approximated errors can be determined for the procedure with

$$e_1 \leq ae_1 = \sum_{i\in \bar{S}_p} \min_{\substack{j\in\mathcal{S}\\ j\neq i}} \left[1 - \prod_{k=1}^{H} P(\bar{X}_{jk} \leq \bar{X}_{ik})\right] \tag{3.17}$$

and

$$e_2 \leq ae_2 = \sum_{i\in S_p} \left(1 - \prod_{\substack{j\in\mathcal{S}\\ j\neq i}} \left[1 - \prod_{k=1}^{H} P(\bar{X}_{jk} \leq \bar{X}_{ik})\right]\right). \tag{3.18}$$

It must be noted that individually the errors are probabilities but in (3.17) and (3.18) the probabilities are summed for S_p and $\bar{S}_p$ for the purpose of the procedure. In other words, the approximated errors themselves are not probabilities but functions of probability; they may therefore add to values greater than one.

3.3.1.3 Formulation of the MOCBA problem

In this section, the MOCBA problem is put forth with its objectives so as to form the allocation rule accordingly. The different formulations by Chen & Lee (2010) and Lee *et al.* (2010b) will be presented.

Lee *et al.* (2010b) formulate the MORS problem as an optimisation problem with the objective of minimising the Type I and Type II errors, given a limited computing budget R_T. This is formulated as

$$\min_{R_1,R_2,\ldots,R_N} ae_1 \tag{3.19}$$

$$\text{s.t.} \sum_{i=1}^{N} R_i \leq R_T, \qquad R_i \geq 0, i = 1, 2, \ldots, N \tag{3.20}$$

and

$$\min_{R_1,R_2,\ldots,R_N} ae_2 \tag{3.21}$$

$$\text{s.t.} \sum_{i=1}^{N} R_i \leq R_T, \qquad R_i \geq 0, i = 1, 2, \ldots, N. \tag{3.22}$$

Lee *et al.* (2010b) present an alternative formulation to the problem whereby allocation rules are derived by minimising the computing budget spent, such that the Type I and Type II errors are within a predefined error limit ε^*. This is formulated as

$$\min \sum_{i=1}^{N} R_i \tag{3.23}$$

$$\text{s.t. } ae_1 \leq \varepsilon^*, \tag{3.24}$$

$$ae_2 \leq \varepsilon^*, \tag{3.25}$$

$$R_i \geq 0, i = 1, 2, \ldots, N. \tag{3.26}$$

The first formulation is used to develop the allocation rule by Lee *et al.* (2010b), but both of the formulation's constraints can be seen in the termination conditions for the rule.

According to Chen & Lee (2010) the main objective of the allocation rule is to maximise the probability of correct selection $P(\text{CS})$. In the case of selecting a set, this is defined as the probability that the approximated Pareto optimal set is the Pareto optimal set and it can be expressed as

$$P(CS) = P\left\{\left(\bigcap_{i\in S_p} E_i\right)\bigcap\left(\bigcap_{i\in \bar{S}_p} E_i^c\right)\right\}. \tag{3.27}$$

Specifically, it is the probability that the scenarios in S_p are non-dominated by all other scenarios, and the scenarios in $\bar{S}_p$ are dominated by at least one other scenario. Keeping simulation efficiency in mind, the problem can be stated as determining the optimal number of simulation replications for each scenario so as to maximise the probability of correct selection, subject to a limited computing budget. This is formulated as

$$\max_{R_1,R_2,\ldots,R_N} P(CS) \tag{3.28}$$

$$\text{s.t.} \sum_{i=1}^{N} R_i = R_T, \qquad R_i \geq 0, i = 1, 2, \ldots, N. \tag{3.29}$$

As the $P(\text{CS})$ cannot be explicitly expressed, it is approximated and the objective of the MORS problem becomes to maximise the approximated probability of correct selection.

3.3.1.4 Asymptotic allocation rules

The aim of the allocation rule is to determine how the simulation budget should be allocated among the scenarios. Although the sample mean and sample variance are used to calculate the allocation, it is asymptotic because when the total simulation budget becomes sufficiently large ($R_T \to \infty$), the calculated allocation approximates the theoretical allocation (which is the true allocation). In all following arguments asymptotic convergence is assumed.

Table 3.2: An example of indices j_i and $k^i_{j_i}$.

i	j_i	$k^i_{j_i}$
1	3	1
2	3	2
3	1	2
4	1	1
5	3	1

The allocation rules by Lee *et al.* (2010b) and Chen & Lee (2010) are based on their respective formulations of the MORS problem (given in Subsection 3.3.1.3). Note that among the allocation rules the author has stated her own concerns, issues and observations about equations where applicable. Where necessary, the author has provided assumptions or simplifications when interpreting the rules.

Lee *et al.* (2010b) and Chen & Lee (2010) define two indices that are used to manipulate the approximated errors and derive the allocation rules. The first index j_i is for the scenario most likely to dominate scenario i and is determined by

$$j_i = \arg\max_{\substack{j\in\mathcal{S}\\ j\neq i}} \prod_{k=1}^{H} P(\bar{X}_{jk} \leq \bar{X}_{ik}) \qquad \text{for } i \in \mathcal{S}. \tag{3.30}$$

The second index, $k^i_{j_i}$ is for the objective of j_i that dominates the corresponding objective of scenario i with the lowest probability. It is determined by

$$k^i_{j_i} = \arg\min_{k\in\mathcal{H}} P(\bar{X}_{j_i k} \leq \bar{X}_{ik}). \tag{3.31}$$

Table 3.2 shows an example of values for the indices. The first row of the table reads, for scenario $i = 1$, $j_1 = 3$ and $k^1_{j_1} = 1$. This means that scenario 3 is most likely to dominate scenario 1, and the objective of scenario 3 that dominates the corresponding objective of scenario 1 with the lowest probability is objective 1.

Using these indices, upper bounds (ub) for the approximated Type I and Type II errors are defined as

$$e_1 \leq ae_1 \leq ub_1 = H|\bar{S}_p| - H\sum_{i\in\bar{S}_p} P(\bar{X}_{j_i k^i_{j_i}} \leq \bar{X}_{ik^i_{j_i}}) \tag{3.32}$$

and

$$e_2 \leq ae_2 \leq ub_2 = (N-1)\sum_{i \in S_p} P(\bar{X}_{j_i k^i_{j_i}} \leq \bar{X}_{i k^i_{j_i}}). \tag{3.33}$$

The explanation of the bounds and how the indices are used is as follows. For ub_1, take a scenario i in $\bar{S}_p$ with the aim to establish that it will be dominated by some other scenarios. Determine which scenario dominates scenario i with the highest probability — this is scenario j_i. When j_i is known, it is only necessary to consider the objective of j_i that dominates the corresponding objective of scenario i the least — this is objective $k^i_{j_i}$. If the probability that scenario j_i is better than scenario i is close to one, then it is known that scenario i belongs in $\bar{S}_p$, as it is dominated by at least one other scenario (j_i). In the same way for ub_2, take a scenario i in S_p, the aim is to establish that it is non-dominated. Determine which scenario dominates scenario i the least — this is scenario j_i. When j_i is known, find the objective of j_i that dominates the corresponding objective of scenario i the most — this is objective $k^i_{j_i}$. If this probability is close to zero, then it is known that scenario i belongs in S_p, as it is not dominated by any other scenario. The relationships between the scenarios in the upper bounds are used to form the asymptotic allocation rules (Lee *et al.*, 2010b).

The allocation rules according to Lee *et al.* (2010b) are presented below. The upper bound of the approximated errors are asymptotically minimised when a proportion of R_T is allocated to scenario i. The proportion α_i is calculated as,

$$\alpha_i = \frac{\beta_i}{\sum_{b \in \mathcal{S}} \beta_b} \qquad \text{for } i \in \mathcal{S}. \tag{3.34}$$

There are two rules to allocate simulation replications to scenarios. Each one focuses on minimising a type of selection error. Which rule is chosen to be applied depends on the values of the Type I and Type II errors at that stage of the procedure.

The rule to minimise the upper bound of the Type I error is as follows. For any scenario $l \in \bar{S}_p$,

$$\beta_l = \frac{\left(\sigma^2_{l k^l_{j_l}} + \sigma^2_{j_l k^l_{j_l}}/\rho_l\right)/\delta^2_{l j_l k^l_{j_l}}}{\left(\sigma^2_{m k^m_{j_m}} + \sigma^2_{j_m k^m_{j_m}}/\rho_m\right)/\delta^2_{m j_m k^m_{j_m}}}, \tag{3.35}$$

Table 3.3: An example of the scenarios in Ω_d.

i	3	5	6	7	9	10
j_i	4	1	10	1	1	4

given that m is any fixed scenario in $\bar{S}_p$ and

$$\rho_i = \frac{\alpha_{j_i}}{\alpha_i}. \tag{3.36}$$

At the first iteration, α_i and therefore ρ_i are unknown so their values are determined by iteratively searching for them. Initially, assume that scenarios i and j_i have been allocated α'_i and α'_{j_i} fractions of simulation replications. Then approximate ρ_i by $\rho_i = \alpha'_{j_i}/\alpha'_i$ and determine α_i for all scenarios. Based on the new allocation, ρ_i can be re-calculated for $i \in \bar{S}_p$. If it is different from the previous value, the process is repeated until ρ_i converges.

For any scenario $d \in S_p$,

$$\beta_d = \sqrt{\sum_{i \in \Omega_d} \frac{\sigma^2_{dk^i_d}}{\sigma^2_{ik^i_d}} \beta_i^2}, \tag{3.37}$$

given that $\Omega_d = \{\text{scenario } i \mid i \in \bar{S}_p \text{ and } j_i = d\}$.

An example of what is in an Ω_d set is now presented. Suppose there are 10 scenarios that are ranked as $S_p = \{1, 2, 4, 8\}$ and $\bar{S}_p = \{3, 5, 6, 7, 9, 10\}$. If we were at the start of calculating β_d, we would take the first scenario in S_p which is $d = 1$. Table 3.3 shows the scenarios in $\bar{S}_p$ and their respective j_i values. For $d = 1$ and the scenarios in $\bar{S}_p$, $j_i = 1$ at scenarios 5, 7 and 9. Therefore $\Omega_1 = \{5, 7, 9\}$.

The rule to minimise the upper bound of the Type II error is as follows. For any scenario $l \in S^A_p$

$$\beta_l = \frac{\left(\sigma^2_{lk^l_{j_l}} + \sigma^2_{j_l k^l_{j_l}}/\rho_l\right)/\delta^2_{lj_i k^l_{j_l}}}{\left(\sigma^2_{mk^m_{j_m}} + \sigma^2_{j_m k^m_{j_m}}/\rho_m\right)/\delta^2_{mj_m k^m_{j_m}}}, \tag{3.38}$$

given that m is any fixed scenario in S^A_p. For any scenario $u \in S^B_p \cup \bar{S}_p$,

$$\beta_u = \sqrt{\sum_{i \in \Theta^*_u} \frac{\sigma^2_{uk^i_u}}{\sigma^2_{ik^i_u}} \beta_i^2}, \tag{3.39}$$

given that $\Theta_d = \{\text{scenario } i \mid i \in S_p \text{ and } j_i = d\}$,

$$S_p^A = \left\{ \text{scenario } l \in S_p \left| \frac{\delta^2_{lj_l k^l_{j_l}}}{\sigma^2_{lk^l_{j_l}}/\alpha_l + \sigma^2_{j_l k^l_{j_l}}/\alpha_{j_l}} < \min_{i \in \Theta_l} \frac{\delta^2_{ilk^i_l}}{\sigma^2_{ik^i_l}/\alpha_i + \sigma^2_{lk^i_l}/\alpha_l} \right. \right\}, \quad (3.40)$$

$$S_p^B = S_p \setminus S_p^A, \text{ and} \quad (3.41)$$

$$\Theta_u^* = \{\text{scenario } i \mid i \in S_p^A \text{ and } j_i = u\}. \quad (3.42)$$

Lee *et al.* (2010b) note that if $\Theta_u^* = \varnothing$ then $\alpha_u = 0$ and similarly, if $\Omega_d = \varnothing$ then $\alpha_d = 0$. This will occur in the second case, if $d \in S_p$, $i \in \bar{S}_p$ and there is no $j_i = d$.

Lee *et al.* (2010b) and Chen & Lee (2010) explain that there are two main types of rules, depending on the role a scenario occupies:

1. Scenarios that are dominated by other scenarios follow the ratio rule where the allocation is directly proportional to the noise to signal ratio.

2. Scenarios that dominate other scenarios follow the square root rule where the allocation is the square root of the sum of weighted squares for those scenarios that it dominates.

According to Lee *et al.* (2010b) for (3.35) and (3.37), scenarios in S_p play the role of dominating other scenarios and scenarios in $\bar{S}_p$ play the role of being dominated. Additionally, in (3.38) and (3.39), due to more than one scenario playing a dominating role (Pareto set), the scenarios in S_p can play both the roles of being dominated by and dominating other scenarios. This is due to the stochastic nature of the simulation. For example, a scenario might be dominated by one scenario with high probability, while also dominating others with a high probability. As a result, a rule is set to establish which roles the scenarios are playing as defined in (3.40). Lee *et al.* (2010b) state that scenarios in S_p^A play the role of being dominated and the scenarios in S_p^B play the role of dominating other scenarios. As the S_p^A set is only chosen from non-dominated scenarios (S_p) it is difficult to see how this is possible. Chen & Lee (2010) give further insight into (3.40). The left side of the inequality is the ratio for the scenario to play

the role of being dominated. The right side of the inequality is the ratio for the scenario to play the role of dominating. Furthermore, Chen & Lee (2010) state that the smaller the signal to noise ratio — on the left-hand side of the inequality in (3.40)— the more significant its role in determining the probability of correct selection. To maximise this probability, more replications are allocated to important scenarios.

Lee *et al.* (2010b) compare MOCBA to the uniform computing budget allocation algorithm (UCBA) which allocates the same number of replications to each scenario. MOCBA is also compared to theoretical optimal allocation (TOA), where the true mean and variance of the performance measures are known in advance and replications are allocated to the scenarios so that both approximated types of errors are minimised. Performance evaluation, for the comparison, is based on the estimated true Type I and Type II errors and the estimated true $P(\text{CS})$. Results show that MOCBA has an improved $P(\text{CS})$ compared to the other algorithms. In addition, both types of errors are lower for MOCBA, and it requires less computing budget to reach the same $P(\text{CS})$, compared to UCBA.

This version of the MOCBA algorithm is presented as background to the MOCBA procedure. Versions of the algorithm that are more straightforward, proposed by Chen & Li (2010), are presented next. The last procedure is used during experimentation.

The allocation rule by Chen & Lee (2010) consists of similar equations to (3.38) to (3.42), with minor changes to the sets. The allocation rule does not use the approximated Pareto and non-Pareto sets and their Type I and Type II errors, instead it only uses the roles the scenarios are playing (dominating or being dominated), determined by an equation similar to (3.40). This already makes the allocation rules simpler than the ones provided by Lee *et al.* (2010b). The allocation rule is now presented.

The approximated probability of correct selection can be asymptotically maximised when

$$\alpha_i = \frac{\beta_i}{\sum_{b \in \mathcal{S}} \beta_b} \qquad \text{for } i \in \mathcal{S}, \tag{3.43}$$

where for any scenario $h \in S^A$

$$\beta_h = \frac{(\sigma^2_{hk^h_{j_h}} + \sigma^2_{j_h k^h_{j_h}}/\rho_h)/\delta^2_{hj_h k^h_{j_h}}}{(\sigma^2_{mk^m_{j_m}} + \sigma^2_{j_m k^m_{j_m}}/\rho_m)/\delta^2_{mj_m k^m_{j_m}}} \tag{3.44}$$

given that m is any fixed scenario in S^A, and for any scenario $d \in S^B$

$$\beta_d = \sqrt{\sum_{i \in \Theta^*_d} \frac{\sigma^2_{dk^i_d}}{\sigma^2_{ik^i_d}} \beta^2_i}. \tag{3.45}$$

Furthermore,

$$j_i = \arg\max_{\substack{j \in \mathcal{S} \\ j \neq i}}, \prod_{k=1}^{H} P(\bar{X}_{jk} \leq \bar{X}_{ik}) \tag{3.46}$$

$$k^i_{j_i} = \arg\min_{k \in \mathcal{H}} P(\bar{X}_{j_i k} \leq \bar{X}_{ik}), \tag{3.47}$$

$$S^A = \left\{ \text{scenario } h \in \mathcal{S} \;\middle|\; \frac{\delta^2_{hj_h k^h_{j_h}}}{\sigma^2_{hk^h_{j_h}}/\alpha_h + \sigma^2_{j_h k^h_{j_h}}/\alpha_{j_i}} < \min_{i \in \Theta_h} \frac{\delta^2_{ihk^i_h}}{\sigma^2_{ik^i_h}/\alpha_i + \sigma^2_{hk^i_h}/\alpha_h} \right\}, \tag{3.48}$$

$$S^B = \mathcal{S} \setminus S^A, \tag{3.49}$$

$$\Theta_h = \{\text{scenario } i \mid i \in \mathcal{S} \text{ and } j_i = h\}, \tag{3.50}$$

$$\Theta^*_d = \{\text{scenario } h \mid h \in S^A \text{ and } j_h = d\} \tag{3.51}$$

and

$$\rho_i = \alpha_{j_i}/\alpha_i. \tag{3.52}$$

Chen & Lee (2010) continue to simplify the allocation rules by excluding the use of ρ_i. This simplification, named MOCBA_simplified, is based on approximations such as ignoring the effect of the allocation when determining the role of

dominating or being dominated. The simplified rule to determine the allocation quantity is as follows. For any scenario $h \in S^A$

$$\beta_h = \left(\frac{\sigma^2_{hk^h_{j_h}} / \delta_{hj_hk^h_{j_h}}}{\sigma^2_{mk^m_{j_m}} / \delta_{mj_mk^m_{j_m}}} \right)^2 \tag{3.53}$$

given that m is any fixed scenario in S^A, and for $d \in S^B$

$$\beta_d = \sqrt{\sum_{h \in \Theta^*_d} \frac{\sigma^2_{dk^h_d}}{\sigma^2_{hk^h_d}} \beta^2_h}, \tag{3.54}$$

where

$$k^i_j = \arg\min_{k \in \mathcal{H}} P(\bar{X}_{jk} \leq \bar{X}_{ik}) = \arg\max_{k \in \mathcal{H}} \frac{\delta_{ijk}|\delta_{ijk}|}{\sigma^2_{ik} + \sigma^2_{jk}} \tag{3.55}$$

and

$$j_i = \arg\max_{\substack{j \in \mathcal{S} \\ j \neq i}} \prod_{k=1}^{H} P(\bar{X}_{jk} \leq \bar{X}_{ik}) = \arg\min_{\substack{j \in \mathcal{S} \\ j \neq i}} \frac{\delta_{ijk^i_j}|\delta_{ijk^i_j}|}{\sigma^2_{ik^i_j} + \sigma^2_{jk^i_j}} \qquad \text{for } i \in \mathcal{S}. \tag{3.56}$$

The indices are different from the ones in the standard rule in Chen & Lee (2010) and Lee *et al.* (2010b) in that in this version $k^i_{j_i}$ does not depend on j_i but the opposite is true. Table 3.4 and Table 3.5 provide examples of the k^i_j and j_i indices for a problem with five scenarios and two objectives. Suppose $i = 1$ and we were to determine $k^1_{j_1}$, which is the objective of j_1 that dominates the corresponding objective of scenario 1 with the lowest probability. We would look at the first row of Table 3.4 for i and the columns for j_i. To determine which column to look at we use the first row of Table 3.5 — this is $j_i = 2$. Therefore we would look at the second column of Table 3.4 and find $k^1_{j_1} = 1$.

Furthermore,

$$S^A = \left\{ \text{scenario } h \in \mathcal{S} \;\middle|\; \frac{\delta^2_{hj_hk^h_{j_h}}}{\sigma^2_{hk^h_{j_h}} + \sigma^2_{j_hk^h_{j_h}}} < \min_{i \in \Theta_h} \frac{\delta^2_{ihk^i_h}}{\sigma^2_{ik^i_h} + \sigma^2_{hk^i_h}} \right\}, \tag{3.57}$$

$$S^B = \mathcal{S} \setminus S^A, \tag{3.58}$$

$$\Theta_h = \{\text{scenario } i \mid i \in \mathcal{S} \text{ and } j_i = h\}, \tag{3.59}$$

Table 3.4: An example of index k_j^i.

	j				
i	1	2	3	4	5
1	2	1	1	1	1
2	2	2	1	1	1
3	2	2	2	1	2
4	2	2	2	2	2
5	2	2	1	1	2

Table 3.5: An example of index j_i.

i	1	2	3	4	5
j_i	2	4	4	3	4

and

$$\Theta_d^* = \{\text{scenario } h \mid h \in S^A \text{ and } j_h = d\}. \tag{3.60}$$

To apply the simplified rule, evaluate (3.55) to (3.60), then compute the allocation quantity (3.43) with (3.53) and (3.54). An iterative procedure is no longer required for the convergence of ρ_i. Chen & Lee (2010) experiment with the MOCBA and MOCBA_simplified algorithms and conclude that MOCBA_simplified may be easier to apply in practice and does not result in a significant compromise in performance.

3.3.1.5 Independence among performance measures

In their work, Lee *et al.* (2010b) acknowledge that in real-life problems, performance measures from the same system are often dependent on each other. Despite this, when deriving MOCBA they assume independence among performance measures to make the problem manageable. They explain that assuming this does not have a significant impact on the algorithm. The reasons are as follows. The allocation rules were derived from the upper bounds of e_1 and e_2 which in turn were derived from the Bonferroni inequality. This is applicable to problems with both independent and correlated objectives. However there is another issue, the approximated errors are used to determine which allocation rule should be applied

(if $ae_1 < ae_2$ minimise Type II error, else minimise Type I error), and this is only valid for problems with independent objectives. To determine if this affects the performance of MOCBA significantly, Lee *et al.* (2010b) conducted a numerical study. Specifically the way MOCBA allocates replications was investigated. The results show that MOCBA allocates replications in a similar way for problems with correlated objectives and for problems with independent objectives. Therefore, correlation does not have a considerable impact on the performance of the algorithm and the assumption of independent objectives is not required.

3.3.1.6 MOCBA sequential procedure

Based on the allocation rules, a heuristic iterative procedure to implement MOCBA by Lee *et al.* (2010b) is shown as Algorithm 4. The simulation budget is allocated in a sequential fashion so as to use the most recent and accurate sample mean and variance information (Lee *et al.*, 2010b). The procedure begins by performing an initial number of replications for all the scenarios. Thereafter at each iteration Δ simulation replications are added to be distributed among scenarios. The simulation output should include the sample mean and variance for each objective of the scenarios. Using this information, S_p is constructed. Subsequently, the errors are calculated to determine the quality of the constructed S_p. Based on this, the appropriate allocation rule is used to allocate a fraction of Δ to each scenario. For example, the number of additional replications to run for scenario i is $\omega_i = \Delta \times \alpha_i$. The procedure continues iteratively, until the stopping criteria have been met. Possible termination conditions are (i) the maximum simulation budget has been reached and (ii) the Type I and Type II errors are within predefined limits.

A heuristic iterative procedure to implement the MOCBA algorithm by Chen & Lee (2010) is shown as Algorithm 5. For the remainder of the study the MOCBA_simplified algorithm that was applied is just referred to as MOCBA.

Algorithm 4 MOCBA algorithm (Lee *et al.*, 2010b)

1: *Input*: Number of scenarios N, maximum total computing budget R_T, maximum additional replications to allocate τ, incremental number of replications Δ.
2: *Initialise:* Perform n_0 replications for each scenario, set the iteration index $v = 0$ and set the individual sample sizes $R_1^v, R_2^v, \ldots, R_N^v = n_0$.
3: *Loop:* **While** (termination condition is not satisfied)
4: *Update:* Construct the observed Pareto set S_p according to (3.7). Calculate the errors e_1 and e_2 according to (3.17) and (3.18).
5: *Allocate:* **If** $ae_1 \leq ae_2$, Calculate the new allocation (3.34) to minimise the Type II error by using (3.38) to (3.42).
6: **Else** Calculate the new allocation (3.34) to minimise the Type I error by using (3.35) to (3.37).
7: *Simulate:* Perform an additional $\min\{\tau, \max(0, R_i^{v+1} - R_i^v)\}$ replications for scenarios $i = 1, 2, \ldots, N$ and set the iteration index $v = v + 1$.

Algorithm 5 MOCBA_simplified algorithm (Chen & Lee, 2010)

1: *Input*: Number of scenarios N, maximum total computing budget R_T, maximum additional replications to allocate τ, incremental number of replications Δ.
2: *Initialise:* Perform n_0 replications for each scenario, set the iteration index $v = 0$ and set the individual sample sizes $R_1^v, R_2^v, \ldots, R_K^v = n_0$.
3: *Loop:* **While** (termination condition is not satisfied)
4: *Update:* Construct the observed Pareto set.
5: *Allocate:* Increase the computing budget by Δ and calculate the new allocation according to (3.53) to (3.60).
6: *Simulate:* Perform an additional $\min\{\tau, \max(0, R_i^{v+1} - R_i^v)\}$ replications for scenarios $i = 1, 2, \ldots, N$ and set the iteration index $v = v + 1$.

3.3.2 Multi-objective optimal computing budget allocation algorithm with indifference-zone framework

In the event that performance measures have means that are very close to one another, it can be tough to differentiate between scenarios. This is due to the observed performance measures being random variables and the closer the true performances of the scenarios, the higher the probability that the observed better scenario is not actually the better one. As a result, a large number of simulation replications are needed to differentiate between those scenarios. However, in practice, stakeholders may not be very concerned with small differences in scenarios and can thus regard the scenarios as being equal. From this stemmed the need to integrate an IZ concept into R&S procedures (Teng *et al.*, 2010). Several solution approaches to R&S use the IZ concept as discussed in Subsection 3.2.1.1.

Scenarios with similar performances in the single-objective case can be treated as equal, but in MORS it is more complicated. For example, when some objectives have true performances that are very close, these objectives should be regarded as indifferent, and the dominance relationship between the scenarios should be established using the remaining objectives. In another situation, when the performance measures of all objectives are very close, the scenarios should be regarded as indifferent and thus not dominated by one another.

Teng *et al.* (2010) examine how to combat the issues regarding systems with close performance measures in the MORS problem. They develop the MOCBA_IZ procedure which integrates the IZ concept into the MOCBA framework. Their work focuses on determining the probability of non-dominance, defining the Pareto optimal set and deriving allocation rules for the replications — now that an IZ has been introduced. For the proof and derivations of the equations used in the procedure, the reader is referred to Teng *et al.* (2010).

The problem considered by the MOCBA_IZ can be stated as follows. Suppose one has scenarios $i \in \mathcal{S} = \{1, \ldots, N\}$, each estimated by independent objectives $k \in \mathcal{H} = \{1, \ldots, H\}$. Each performance measure is represented by a random variable following a normal distribution, with true mean μ_{ik} and finite variance σ_{ik}^2. Given a specified IZ for each objective, the aim is to determine the optimal allocation of replications to the scenarios so that the non-dominated set can

be found with high confidence and the smallest simulation budget. Given that R_i simulation observations $(X_{ik}^1, X_{ik}^2, \ldots X_{ik}^{R_i})$ are taken for the k-th objective of scenario i, the sample mean performance is $\bar{X}_{ik} = \frac{\sum_{a=1}^{R_i} X_{ik}^a}{R_i}$. The probability distribution of $\bar{X}_{ik}$ (the sampling distribution of μ_{ik}) is

$$\bar{X}_{ik} \sim N\left(\mu_{ik}, \frac{\sigma_{ik}^2}{R_i}\right). \tag{3.61}$$

3.3.2.1 Dominance relationship with indifference-zone and its probability description based on observed performance

When an IZ is taken into account, the traditional definition of the dominance relationship and the Pareto optimal set needs to be redefined. Let δ_k^* be the IZ chosen for objective k. Consider two scenarios i and j. Scenario j dominates scenario i, denoted by $j \hat{\prec}_{\text{IZ}} i$ (by observation) when the following conditions hold:

$$\begin{aligned} &\text{for every objective } k \in \mathcal{H}, \bar{X}_{jk} \lesssim_{\text{IZ}} \bar{X}_{ik} \qquad (\bar{\delta}_{ijk} \leq \delta_k^*) \\ &\text{and for at least one } k \in \mathcal{H}, \bar{X}_{jk} <_{\text{IZ}} \bar{X}_{ik} \qquad (\bar{\delta}_{ijk} < -\delta_k^*). \end{aligned} \tag{3.62}$$

The first condition determines whether for all objectives, scenarios are within the IZ from each other. If this is true then the first condition is met. The second condition ensures that in at least one objective, the scenarios do not have very close performances (larger than the IZ) and the one scenario is significantly better than the other scenario.

Using the conditions in (3.62), S_p with IZ can be constructed by placing scenario i in the approximated Pareto set such that for $j \hat{\prec} i$ a scenario j cannot be found. Teng *et al.* (2010) further discuss how the introduction of the IZ changes the traditional meaning of the dominance relationship and how there can be differences in S_p and S_p with IZ, by means of two examples. Approximated Pareto set construction is implemented by incorporating the IZ into the Pareto ranking algorithm by (Goldberg, 1989). The modified algorithm is presented in Appendix B.

The probability that scenario j dominates scenario i is calculated by

$$P(j \hat{\prec}_{\text{IZ}} i) = \prod_{k=1}^{H} \Phi(Z_{ijk}^*) - \prod_{k=1}^{H} (\Phi(Z_{ijk}^*) + \Phi(Z_{jik}^*) - 1) \tag{3.63}$$

where

$$Z^*_{ijk} = \frac{\delta^*_{ijk}}{\sqrt{\sigma^2_{ik}/R_i + \sigma^2_{jk}/R_j}}, \qquad Z^*_{jik} = \frac{\delta^*_{jik}}{\sqrt{\sigma^2_{ik}/R_i + \sigma^2_{jk}/R_j}}, \tag{3.64}$$

and where

$$\delta^*_{ijk} = \delta^*_k - \delta_{ijk} = \begin{cases} \delta^*_k & \text{if } |\delta_{ijk}| \leq \delta^*_k \\ -\delta_{ijk} & \text{otherwise} \end{cases} \tag{3.65}$$

and

$$\delta^*_{jik} = \delta^*_k - \delta_{jik} = \begin{cases} \delta^*_k & \text{if } |\delta_{ijk}| \leq \delta^*_k \\ -\delta_{jik} & \text{otherwise .} \end{cases} \tag{3.66}$$

The explanation of (3.65) and (3.66) follows that when objectives of scenarios are very close ($|\delta_{ijk}| \leq \delta^*_k$), they should be classified as indifferent so that the relationship between the scenarios i and j can be determined by the remaining objectives. To do this, δ_{ijk} is set to zero ($\delta_{ijk} = 0$). Alternatively, when scenarios have objective function values that are very different ($|\delta_{ijk}| > \delta^*_k$), the IZ needs not be considered and δ^*_k is set to zero ($\delta^*_k = 0$).

3.3.2.2 Two types of errors of the approximated Pareto set

In the same way as in Lee *et al.* (2010b), Teng *et al.* (2010) give a Type I and Type II error, that incorporates IZ, to evaluate the quality of the approximated Pareto set. The two types of errors are given by

$$\begin{aligned} e_1 \leq ae_1 &= \sum_{i \in \bar{S}_p} \min_{\substack{j \in \mathcal{S} \\ j \neq i}} [1 - P(j \hat{\prec} i)] \\ &= \sum_{i \in \bar{S}_p} \min_{\substack{j \in \mathcal{S} \\ j \neq i}} \left[1 - \prod_{k=1}^{H} \Phi(Z^*_{ijk}) + \prod_{k=1}^{H} (\Phi(Z^*_{ijk}) + \Phi(Z^*_{jik}) - 1) \right] \end{aligned} \tag{3.67}$$

and

$$\begin{aligned} e_2 \leq ae_2 &= \sum_{i \in S_p} (1 - P(j \hat{\prec} i)) \\ &= \sum_{i \in S_p} \left(1 - \prod_{\substack{j \in \mathcal{S} \\ j \neq i}} \left[1 - \prod_{k=1}^{H} \Phi(Z^*_{ijk}) + \prod_{k=1}^{H} (\Phi(Z^*_{ijk}) + \Phi(Z^*_{jik}) - 1) \right] \right). \end{aligned} \tag{3.68}$$

3.3.2.3 Asymptotic allocation rules

Teng *et al.* (2010) formulate the MORS problem as an optimisation problem with the objective of minimising the Type I and Type II errors, given a limited computing budget. The allocation rules are then derived from this and are given as follows. Two indices used in the rules are

$$\begin{aligned} j_i &= \arg\max_{\substack{j\in\mathcal{S}\\ j\neq i}} P(j\hat{\prec}i) \\ &= \arg\max_{\substack{j\in\mathcal{S}\\ j\neq i}} \left(\prod_{k=1}^{H} \Phi(Z^*_{ijk}) - \prod_{k=1}^{H} (\Phi(Z^*_{ijk}) + \Phi(Z^*_{jik}) - 1) \right) \end{aligned} \tag{3.69}$$

and

$$k^i_{j_i} = \arg\min_{k\in\mathcal{H}} \Phi(Z^*_{ij_ik}). \tag{3.70}$$

The fraction of the total number of simulation replications to be allocated to scenario i is

$$\alpha_i = \frac{\beta_i}{\sum_{b\in\mathcal{S}} \beta_b} \qquad \text{for } i \in \mathcal{S}. \tag{3.71}$$

To minimise the upper bound of the Type I error, the following set of equations is used. For any scenario $l \in \bar{S}_p$ and given that m is any fixed scenario in $\bar{S}_p$,

$$\beta_l = \frac{\left(\sigma^2_{lk^l_{j_l}} + \sigma^2_{j_lk^l_{j_l}}/\rho_l\right)/(\delta^*_{lj_ik^l_{j_l}})^2}{\left(\sigma^2_{mk^m_{j_m}} + \sigma^2_{j_mk^m_{j_m}}/\rho_m\right)/(\delta^*_{mj_mk^m_{j_m}})^2} \tag{3.72}$$

where

$$\rho_i = \frac{\alpha_{j_i}}{\alpha_i} \tag{3.73}$$

and for any scenario $d \in S_p$

$$\beta_d = \sqrt{\sum_{i\in\Omega_d} \frac{\sigma^2_{dk^i_d}}{\sigma^2_{ik^i_d}} \beta^2_i}, \tag{3.74}$$

given that $\Omega_d = \{\text{scenario } i \mid i \in \bar{S}_p \text{ and } j_i = d\}$. The value of ρ_i is initially assumed and then iteratively determined until it converges, as in MOCBA.

To minimise the upper bound of the Type II error, the following set of equations is used. For any scenario $l \in S_p^A$ and given that m is any fixed scenario in S_p^A,

$$\beta_l = \frac{\left(\sigma^2_{lk^l_{j_l}} + \sigma^2_{j_l k^l_{j_l}}/\rho_l\right)/(\delta^*_{lj_ik^l_{j_l}})^2}{\left(\sigma^2_{mk^m_{j_m}} + \sigma^2_{j_m k^m_{j_m}}/\rho_m\right)/(\delta^*_{mj_mk^m_{j_m}})^2} \tag{3.75}$$

and for any scenario $u \in S_p^B \cup \bar{S}_p$,

$$\beta_u = \sqrt{\sum_{i\in\Theta^*_u} \frac{\sigma^2_{uk^i_u}}{\sigma^2_{ik^i_u}}\beta_i^2} \tag{3.76}$$

given that $\Theta_d = \{\text{scenario } i \mid i \in S_p \text{ and } j_i = d\}$, where

$$S_p^A = \left\{ \text{scenario } l \in S_p \middle| \frac{(\delta^*_{lj_ik^l_{j_l}})^2}{\sigma^2_{lk^l_{j_l}}/\alpha_l + \sigma^2_{j_lk^l_{j_l}}/\alpha_{j_l}} < \min_{i\in\Theta_l} \frac{(\delta^*_{ilk^i_l})^2}{\sigma^2_{ik^i_l}/\alpha_i + \sigma^2_{lk^i_l}/\alpha_l} \right\}, \tag{3.77}$$

$$S_p^B = S_p \setminus S_p^A, \tag{3.78}$$

and

$$\Theta^*_d = \{\text{scenario } i \mid i \in S_p^A \text{ and } j_i = d\}. \tag{3.79}$$

3.3.2.4 MOCBA_IZ sequential procedure

The procedure to allocate the simulation budget for a MORS problem with IZ by Teng *et al.* (2010) is given in Algorithm 6.

3.3.2.5 Comparison of MOCBA_IZ and MOCBA

Teng *et al.* (2010) experimented with the MOCBA_IZ algorithm to determine its performance compared to the MOCBA and UCBA algorithm, when applied to a problem containing scenarios with close performances among some objectives. Only the MOCBA and MOCBA_IZ comparison is of interest in this study. They conclude that MOCBA_IZ has a significantly higher probability of correctly selecting the Pareto optimal set, $P(\text{CS})$, and as the simulation budget increases, the improvement of $P(\text{CS})$ increases. The results also show that as the simulation

Algorithm 6 MOCBA_IZ algorithm

1: *Input:* Number of scenarios N, maximum total computing budget R_T, maximum additional replications to allocate τ, incremental number of replications Δ and indifference-zone δ_k^*.
2: *Initialise:* Perform n_0 replications for each scenario, set the iteration index $v = 0$ and set the individual sample sizes $R_1^v, R_2^v, \ldots, R_S^v = n_0$.
3: *Loop:* **While** (termination condition is not satisfied)
4: *Update:* Construct the observed Pareto set S_p according to (3.62) (see Appendix B). Calculate the errors e_1 and e_2 according to (3.67) to (3.68).
5: *Allocate:* **If** $ae_1 \leq ae_2$, calculate the new allocation (3.71) to minimise the Type II error by using (3.75) to (3.79).
6: **Else** Calculate the new allocation (3.71) to minimise the Type I error by using (3.72) to (3.74).
7: *Simulate:* Perform an additional $\min\{\tau, \max(0, R_i^{v+1} - R_i^v)\}$ replications for scenarios $i = 1, 2, \ldots, N$ and set the iteration index $v = v + 1$.

budget increases, MOCBA_IZ has a more rapid decrease in both types of errors, compared to MOCBA. These findings are explained by the way in which the different allocation procedures allocate simulation replications. The replications for MOCBA_IZ are more evenly distributed for scenarios that have objectives within the IZ than that for MOCBA. MOCBA requires many replications to differentiate between two scenarios with close performances, which does not improve its $P(\text{CS})$. By regarding objectives with a difference within the IZ as equal, the scenarios are compared according to the rest of the objectives. Consequently, there is a greater chance to place scenarios into the correct set (Pareto or non-Pareto) so both types of errors decrease and the $P(\text{CS})$ increases.

3.3.2.6 MOCBA_IZ simple version

The MOCBA_IZ algorithm has the same issue regarding iteratively determining ρ_i as the MOCBA algorithm. Therefore, the IZ concept was combined with MOCBA_simplified (the simplified version of MOCBA), which does not make use of the ρ_i value. The MOCBA_IZ simple allocation rules are given as follows.

Two indices used in the rules are

$$k_j^i = \arg\max_{k\in\mathcal{H}} \frac{\delta^*_{ijk}|\delta^*_{ijk}|}{\sigma^2_{ik}+\sigma^2_{jk}} \tag{3.80}$$

and

$$j_i = \arg\min_{\substack{j\in\mathcal{S}\\ j\neq i}} \frac{\delta^*_{ijk^i_j}|\delta^*_{ijk^i_j}|}{\sigma^2_{ik^i_j}+\sigma^2_{jk^i_j}} \qquad \text{for } i\in\mathcal{S}, \tag{3.81}$$

where

$$\delta^*_{ijk} = \delta^*_k - \delta_{ijk} = \begin{cases} \delta^*_k & \text{if } |\delta_{ijk}| \leq \delta^*_k \\ -\delta_{ijk} & \text{otherwise .} \end{cases} \tag{3.82}$$

The fraction of the total number of simulation replications to be allocated to scenario i is

$$\alpha_i = \frac{\beta_i}{\sum_{b\in\mathcal{S}}\beta_b} \qquad \text{for } i\in\mathcal{S}. \tag{3.83}$$

For any scenarios $h, m \in S_A$

$$\beta_h = \left(\frac{\sigma^2_{hk^h_{j_h}}/\delta^*_{hj_hk^h_{j_h}}}{\sigma^2_{mk^m_{j_m}}/\delta^*_{mj_mk^m_{j_m}}}\right)^2 \tag{3.84}$$

and for $d \in S_B$

$$\beta_d = \sqrt{\sum_{h\in\Theta^*_d} \frac{\sigma^2_{dk^h_d}}{\sigma^2_{hk^h_d}}\beta^2_h}, \tag{3.85}$$

where

$$S_A = \left\{ \text{scenario } h\in\mathcal{S} \;\middle|\; \frac{(\delta^*_{hj_hk^h_{j_h}})^2}{\sigma^2_{hk^h_{j_h}}+\sigma^2_{j_hk^h_{j_h}}} < \min_{i\in\Theta_h} \frac{(\delta^*_{ihk^i_h})^2}{\sigma^2_{ik^i_h}+\sigma^2_{hk^i_h}} \right\}, \tag{3.86}$$

$$S_B = \mathcal{S}\setminus S_A, \tag{3.87}$$

$$\Theta_h = \{\text{scenario } i \mid i\in\mathcal{S} \text{ and } j_i = h\}, \tag{3.88}$$

and

$$\Theta^*_d = \{\text{scenario } h \mid h\in S_A \text{ and } j_h = d\}. \tag{3.89}$$

This procedure was used for experimentation. The final MORS procedure is discussed next.

3.3.3 Two-stage-Pareto-set-selection procedure

Chen & Lee (2009) extend the sequential R&S procedure for selecting a subset (see Chen (2007)), by incorporating the IZ approach as well as Pareto ranking to select the non-dominated solutions. This is known as the two-stage-Pareto-set-selection procedure (TSPS). The required sample sizes, that provide a probability guarantee that the scenarios in the approximated Pareto set are non-dominated, are minimised.

As in MOCBA_IZ, this procedure aims to avoid allocating a large number of simulation replications to differentiate between scenarios that have performances within the specified IZ δ_k^*. In addition to the IZ, the decision maker specifies the minimum acceptable probability of correctly selecting the best scenario denoted by P^*. The goal is then to have $P(CS) \geq P^*$, provided that the difference in performance measures of the selected scenario and the next best scenario is greater than δ_k^*. The problem can be stated as follows. Suppose N scenarios exist that are each evaluated by H independent objectives. The aim of the procedure is to determine the non-dominated scenarios by allocating replications so that there is a certain probability guarantee that the identified scenarios are actually non-dominated.

As the name of the procedure suggests, it consists of two stages. During the first stage, an incomplete Pareto set is selected which contains the non-dominated scenarios that are best in at least one performance measure. This stage of the procedure is specifically named the *sequential ranking and selection of an incomplete Pareto set* (SRSIP) procedure. Denote the sample mean $\bar{X}_{ik}$ and sample variance $S_{ik}^2(R_i)$ for scenario $i = 1, 2, \ldots, N$ and objective $k = 1, 2, \ldots, H$, based on R_i observations. Equations for the SRSIP procedure from Chen & Lee (2009) are presented next and the procedure is given in Algorithm 7.

The required sample size of scenario i based on objective k is given by

$$R_{ik,q+1} = \max(n_0 + 1, \lceil (hS_{ik}(R_{i,q})/\delta_k^*)^2 \rceil) \tag{3.90}$$

where n_0 is the initial sample size, δ_k^* is the user specified IZ for the k-th objective and q is the iteration index. Some of the values of the critical constant h can be found in Law & Kelton (2000) or Chen & Li (2010). The value depends on a number of parameters such as P^* and N.

The required sample size for scenario i is

$$R_{i,q+1} = \max_{k=1}^{H} R_{ik,q+1}. \tag{3.91}$$

The t-score is computed by

$$T_k = \frac{\bar{X}_{b_2k} - \bar{X}_{b_1k}}{\sqrt{S^2_{b_2k}(R_{b_2})/R_{b_2} + S^2_{b_1k}(R_{b_1})/R_{b_1}}}. \tag{3.92}$$

Algorithm 7 SRSIP procedure (Chen & Lee, 2009)

1: Input: Specify the value of the indifference amounts δ^*_k, the required precision P^* and $m_p \leq H$, the number of performance measures from which non-dominated scenarios will be selected.
2: Simulate n_0 observations for all scenarios. Set the iteration index $q = 0$, and the individual sample sizes $R_{1,q}, R_{2,q}, \ldots, R_{N,q} = n_0$. Initialise the counter for the number of scenarios that have been declared non-dominated $l = 0$.
3: For each objective $k = 1, 2, \ldots, H$, rank the sample means such that $\bar{X}_{b_1k} \leq \bar{X}_{b_2k} \leq \ldots \leq \bar{X}_{b_Nk}$ and obtain the ranking index b_j for $j = 1, 2, \ldots, N$.
4: Calculate the new sample size $R_{i,q+1}$ according to (3.90) and (3.91), for $i = 1, 2, \ldots, N$.
5: For $k = 1, 2, \ldots H$, calculate the t-score by (3.92).
6: Rank the t-scores such that $T_{k_1} \leq T_{k_2} \leq \ldots \leq T_{k_H}$ for $k = 1, 2, \ldots, H$.
7: Let $P = (P^*)^{1/(N-1)}$, $R_{b_{min}} = \min(R_{b_1}, R_{b_2})$, and let $t_{p,f}$ denote the p-quantile of the t-distribution with f degrees of freedom. For each k such that $T_k > t_{P,R_{b_{min}}-1}$, declare that scenario b_1 is non-dominated with respect to objective k and set $l = l + 1$.
8: Find the objective k_l such that $T_{kl} = \max(T_k | T_k < t_{P,R_{b_{min}}-1}, k = 1, 2, \ldots, H)$.
9: For each objective k, if $R_{i,q+1} \leq R_{i,q}$ for all $i = 1, 2, \ldots, N$, then set $l = l + 1$.
10: If $l \geq m_p$, then go to Step 12.
11: If $R_{i,q+1} - R_{i,q} < 5$, set $\omega_{i,q+1} = [R_{i,q+1} - R_{i,q}]^+$ where $(X)^+ = \max(0, X)$. Else set $\omega_{i,q+1} = 5$. Simulate an additional $\omega_{i,q+1}$ observations for scenario i. Set $R_{i,q+1} = R_{i,q} + \omega_{i,q+1}$ and $q = q + 1$. Go back to Step 3.
12: Return the best scenario of each objective k, for $l = 1, 2, \ldots, m_p$.

The second stage of the TSPS involves determining the complete Pareto set (all the non-dominated scenarios). Let S_{IP} be the current incomplete Pareto set (S_{IP}^c is the complement). Scenarios will be compared to one another to determine the other non-dominated scenarios. The comparison process is as follows. Take $j \in S_{IP}^c$ and compare it with all $i \in S_{IP}$. If it is not dominated by any scenarios in S_{IP} then temporarily designate it as non-dominated. Update S_{IP} by adding scenario j to the S_{IP} set. Repeat this process for all $j \in S_{IP}^c$. After this is complete, perform all-pairwise comparisons among scenarios temporarily classified as non-dominated to remove the dominated scenarios. Figure 3.4 presents a high-level flowchart of the TSPS procedure.

The TSPS procedure falls outside the scope of this study and therefore it was not implemented.

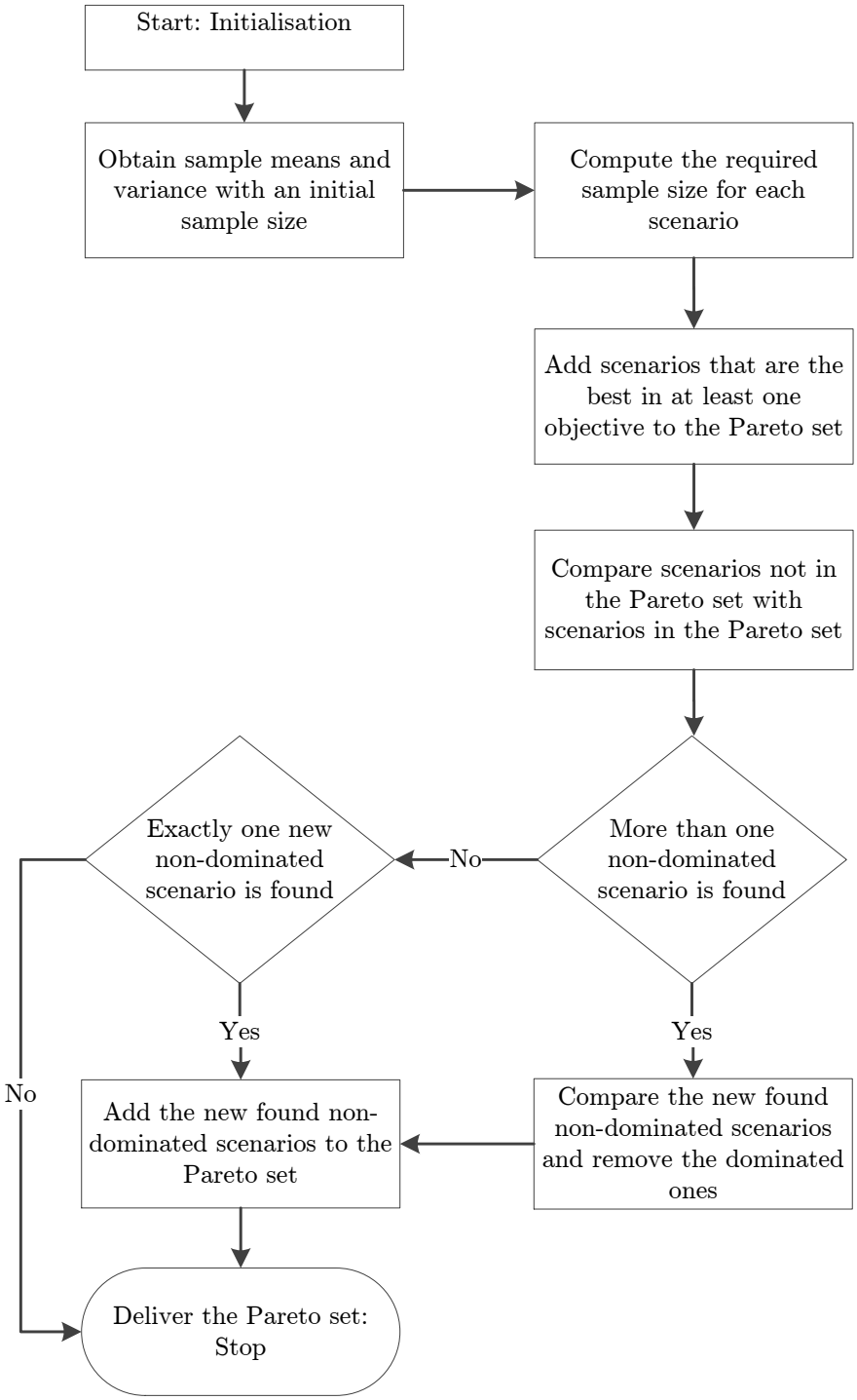


Figure 3.4: Two-stage-Pareto-set-selection procedure (Chen & Lee, 2009).

3.4 Experimental design

This section presents the approach followed for experimentation with MORS. The MORS algorithms are tested on two stochastic problems where computer simulation models provide the objective function set.

3.4.1 Test problems

The focus of this study is not to develop impressive computer simulation models of complex systems. For this reason, two simulation models that were familiar to the researcher, and that had already been validated, were chosen as test problems. These are the buffer allocation problem and a single-commodity inventory problem.

3.4.1.1 The buffer allocation problem

In this section, the finite buffer queuing network problem is described. Finite capacity queueing networks are employed in a number of real-world applications, such as manufacturing systems and telecommunication networks. The flow through the system can either be discrete or continuous. Buffer space is often needed in these networks due to asynchronous part transfer or flow variation.

A priority of network design in this case is to maximise the throughput of the system, which increases with additional buffer space (Cruz *et al.*, 2010). However, buffer space is an expensive resource, so stakeholders desire to minimise this space simultaneously. Through this trade-off, the buffer allocation problem (BAP) emerged. The BAP is of particular concern to operations managers because the allocation of buffer space is one of the limited changes that can be made to a manufacturing system, although plant layout may also be an issue (Papadopoulos *et al.*, 2009).

The BAP is described as follows: a number of buffer slots n are to be allocated among m servers in a network, to meet some objective. The servers are placed sequentially in this study. The buffer locations are placed before each workstation in the network and the buffer in front of the first workstation is infinite. Thus,

$m-1$ buffers exist. Each discrete workpart takes up one buffer slot or occupies a machine and workparts are processed in sequence.

Popular objectives to the BAP include minimising the average work-in-progress in the system, maximising the throughput rate of the production line, minimising the total cost and minimising the total number of buffer slots (Papadopoulos *et al.*, 2009). One of these can be chosen to form a single-objective problem or more than one can be considered as a multi-objective problem. Decision variables for the BAP can include buffer sizes and server processing rates. A buffer may be assigned zero slots, so a workpart occupying a machine can only advance to the next machine if it is operational and unoccupied. At least two versions of the BAP are found in literature, the difference being in the constraint on the allocation of buffers:

1. The total buffer slots allocated must be equal to a predetermined number n. Let B_i be the buffer slot between machine i and $i+1$, $(1 \leq i \leq m-1)$, then $\sum_{i=1}^{m-1} B_i = n$. In this case, the size of the problem is $\binom{m+n-2}{m-2}$.

2. The minimum number of buffer slots are allocated to their respective workstations to optimise one or more objectives. The buffer slots across the network need not be equal. There are $\Pi_{i=1}^{m-1}(n_i+1)$ possible combinations of buffer sizes, and the maximum total number of buffer slots that can be allocated is $\sum_{i=1}^{m-1} n_i$.

Within the versions of the BAP, variations exist where changes are typically made to the types of distributions representing the service time, time-to-failure and repair times, synchronous or asynchronous part transfer, network topology and servers that are reliable or unreliable. A series topology of the BAP is illustrated in Figure 3.5.

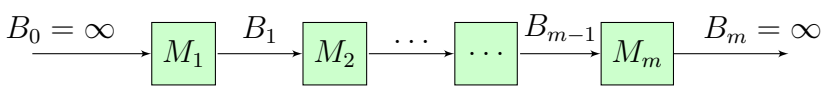


Figure 3.5: A series of machines $M_1, \ldots, M_m$ with finite buffers $B_1, \ldots, B_{m-1}$ in a queuing network.

The BAP is classified as a stochastic, non-linear, integer mathematical programming problem and is therefore computationally hard to solve (Cruz *et al.*, 2008). Papadopoulos & Heavey (1996) classify queueing network models for production and transfer lines. Cruz *et al.* (2008) discuss four solution approaches to the BAP: simulation methods, metaheuristics, dynamic programming and search methods. Bekker (2012) provides an overview of research performed on the BAP over several decades.

The performance measures taken into account in this study are throughput rate $T_R(\mathbf{x})$ (to be maximised) and work-in-progress $W_P(\mathbf{x})$ (to be minimised). The estimated values of the objectives are obtained from a simulation model of the BAP. The formulation for the BAP is as follows

$$\text{Minimise } S_B(\mathbf{x}) := [-T_R(\mathbf{x}), W_P(\mathbf{x})] \tag{3.93}$$

$$\text{subject to } \mathbf{x} \in \mathcal{X},$$

$$\sum_{i=1}^{m-1} x_i = n, \tag{3.94}$$

where $\mathcal{X}$ is the set of all possible combinations of $\mathbf{x}$. From the formulation, one can see that the decisions to be made are the size of the buffers.

Normally the processing and repair times of the machines in the system are exponentially distributed with mean rates μ_i and r_i, respectively. The machines have operation dependent failures, which are more realistic, compared to a time dependent failure type (Yang *et al.*, 2000). For operation based failures, a machine breaks down after it has processed a certain number of products, assigned by Poisson distributions with rates λ_i. When a machine fails, an upstream machine can become blocked and a downstream machine can become starved.

The problem for this study is a manufacturing process consisting of five machines: a mill, punch, lathe, riveting machine and bending machine. The machines have exponentially distributed processing times with rates $\mu_1 = 1$, $\mu_2 = 1.1$, $\mu_3 = 1.2$, $\mu_4 = 1.3$, $\mu_5 = 1.4$, failure rates $\lambda_i = 1/20$ and repair rates $r_i = 0.5$. The simulation run length was set at 500 hours.

The second test problem is presented in the next section.

3.4.1.2 The inventory problem

In this section, a simple dynamic, stochastic inventory problem is described. The problem is similar to the (s, S) policy, a stochastic inventory management model where s refers to the reorder level and S to the reorder quantity. Bashyam & Fu (1998) provide a comprehensive description of the problem.

The specific inventory model considered in this study is as follows: Customers enter the system according to an arrival distribution, in this case the interarrival times are exponentially distributed with a mean of three minutes. Customers demand a discrete product distributed according to:

$$f(x) = \max \begin{cases} 0.2, & x = 2 \\ 0.15, & x = 4 \\ 0.3, & x = 8 \\ 0.22, & x = 13 \\ 0.13, & x = 16. \end{cases} \tag{3.95}$$

The inventory level will decrease by x after the customer request has been successful.

Once the inventory level in the system drops below the reorder level, a specific quantity of products will be reordered. The order lead time (in hours) for products is modelled using a triangular distribution with lower limit, mode and upper limit 1,2,4 respectively. While waiting for this delivery to arrive, customers still purchase the product and the inventory level could reach zero. In this situation, a stock-out period will occur during which customers cannot be served. The demand that customers had during this period is regarded as lost sales, which is undesirable from a profit perspective. However, storing stock incurs a holding cost of ZAR0.06/unit/day, also affecting profit. In addition there is an administration fee associated with each order of ZAR120. An infinite holding area and reliable supplier are assumed.

The service level is the percentage of demand met and is determined by

$$\textit{Service level} = \frac{\textit{Number of customers serviced}}{\textit{Number of customers requiring service}} \times 100\%. \tag{3.96}$$

Note that if a customer desires x units, but the inventory level is below x, the available quantity is given to the customer and the transaction is considered successful. The reorder level s and the reorder quantity S are the decision variables

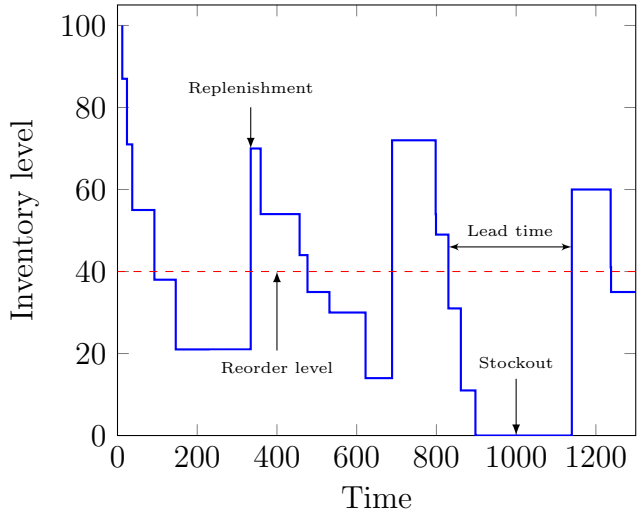


Figure 3.6: Some characteristics of the (s, S) inventory process.

of the problem. The first objective of the multi-objective optimisation problem is to minimise the average total cost which is affected by the number of orders and the carry cost. The second objective is to maximise the service level. The simulation runtime was set to 120 hours. To make the model more realistic, the initial inventory in the system at the beginning of the run time is set as 300 units. An example of the inventory level in the system over time is illustrated in Figure 3.6.

3.4.2 Experimental setup

This section provides details relating to the experiments performed for this part of the study. Six experiments are run. The parameter settings of the algorithms used in the experiments are also provided.

3.4.2.1 Allocation rules for experimentation

Three allocation rules were tested and compared on the two test problems described. The aim is to determine which rule performs best in terms of finding the most optimal Pareto front. This will aid in working towards the ultimate aim of the research, by determining what rule to incorporate into the multi-objective optimisation method using the cross-entropy method (MOO CEM). The first two

rules are MOCBA and MOCBA_IZ. The third rule is the equal allocation (EA) which allocates the same number of simulation replications to each scenario. To determine the number of replications to allocate, the total number of replications given to the other algorithms was taken and divided equally by the number of scenarios. For example, if $R_T = 4\,000$ and $N = 265$ then each scenario receives $4\,000/265 \approx 15$ replications. The equal allocation rule is also known as the uniform computing budget allocation (UCBA) in literature.

3.4.2.2 Algorithm specific parameters

In this section, the initial conditions for the algorithms — MOCBA and MOCBA_IZ are discussed.

Juxin (2012) provides recommendations for the parameters of the MOCBA algorithm. As MOCBA is a sequential procedure, where the subsequent allocation depends on the current (most recent) sampling information, n_0 should not be too small so as to avoid poor initial estimation. The number of replications to be allocated to scenarios per iteration (Δ) should not be too large, to leave potential to correct allocation errors in the next iteration. The value of τ should also not be too large for potential correction of allocation errors. Juxin (2012) recommends the following: $n_0 \geq 5$, $\Delta < 50$ when $N < 20$ and $\tau \leq 10$.

The idea of having a limit on the total number of replications per scenario was investigated. For example, is it worthwhile to perform 1 000 replications on one scenario while less than 100 have been run for all the other scenarios? Surely at some point no additional information will be gained? However, the idea was not implemented in the final experiments so as to not disrupt the algorithm as it was originally published, by perhaps choosing a limit that is too low.

A sensitivity analysis was performed on the parameters of the algorithms. It was concluded that, either changing the parameters did not have a significant effect on the solution (as in the case of the number of points in the Pareto front), or that the changes had a logical effect (a smaller computing budget results in fewer iterations).

For the two problems, three IZ values were tested for each of the two objectives. It was found that the IZ values must be very small, in order to allow

scenarios into the approximated Pareto set, and not be excluded (from the set) by one of the conditions (3.62). In the BAP, the size of the IZ for the work-in-progress (WIP) has the largest effect on the number of scenarios in the approximated Pareto set. In the inventory model, the service level objective has the largest effect. All parameters values tested, together with their results are provided in Appendix C. Base parameters for the experiments were chosen according to these results. The settings chosen for MOCBA using the BAP model are as follows: $R_T = 4\,000$, $n_0 = 5$, $\tau = 20$ and $\Delta = 300$. The same settings are used for MOCBA_IZ with the IZ amounts as 0.005 for the throughput rate and 0.1 for the WIP. In other words, the throughput IZ is 0.5% and the WIP is a 10th of a product. The parameters for MOCBA for the inventory model are: $R_T = 3000$, $n_0 = 5$, $\tau = 20$ and $\Delta = 200$. Additionally, for MOCBA_IZ the IZ for service level is 0.01 (1 %) and for cost is ZAR5. For the EA rule, 15 replications were run for each scenario in the BAP problem, and 16 replications for the inventory problem.

3.4.3 Simulation-optimisation models

The simulation optimisation models are constructed by integrating the simulation models with the optimisation algorithms.

3.4.3.1 Integrating Matlab® and Simio

The MORS algorithms were coded in Matlab® and the discrete-event simulation models of the BAP and inventory problem were developed in the modelling software, Simio. Simio provides an application programming interface (API) which can be used to call Simio models from other programs. However, this is not available for Matlab®. Therefore, to integrate the two components, a `C#` executable and comma separated value files were used to form the simulation-optimisation model. The `C#` executable sets up and runs the Simio model and is called directly from Matlab. The comma separated value files are used to convey data between the programs.

3.4.3.2 Scenarios for evaluation

As MORS is not a search algorithm, a set of possible scenarios for the problem is required prior to experimentation. The MORS algorithm then determines the non-dominated solutions from this set.

The possible scenarios for the BAP were determined by taking all the combinations of integer buffer sizes from 0 to 10 for all the buffer slots ($0 < B_i < 10$). This lead to 14 641 possible combinations. Five simulation replications were run for all these combinations. Subsequently, the Pareto ranking algorithm (Goldberg, 1989) was applied with a threshold of two to rank the scenarios. This yielded 265 scenarios ranked as the elite set. These 265 scenarios were used as the possible scenarios for the MORS algorithms.

The possible scenarios for the inventory model were determined by performing an exhaustive simulation on all the combinations of $10 < s < 1\,000$ and $10 < S < 1\,000$ in increments of 10. Each (s, S) combination was evaluated using 10 simulation replications. Again, Goldberg's ranking algorithm was applied with a threshold of six. Out of the 5 050 combinations simulated, this yielded 186 scenarios. These were used to test the MORS algorithms.

These test sets can be equivalent to a decision maker having a number of possible scenarios for their system and they desire the optimal scenario.

3.4.4 Performance assessment

During experimentation, approximation Pareto sets were generated. Based on these, the performance of each algorithm was determined.

3.4.4.1 A note on probability of correct selection

All MORS algorithms investigated use the $P(\text{CS})$ as a common measure to compare the performance of algorithms. For example, an experiment is repeated 10 000 times and the percentage of cases in which the true Pareto front is found, is calculated as the $P(\text{CS})$. This measures the efficiency of a procedure (Branke *et al.*, 2007). When using stochastic simulation models, the $P(\text{CS})$ cannot be determined because the true Pareto front is unknown. The solutions obtained

from the MORS algorithms are approximations of the true Pareto front of the optimisation problem. For this reason the $P(\text{CS})$ cannot be used to measure performance in this study.

3.4.4.2 Hyperarea indicator

For multi-objective problems, hyperarea (HA) is a simple way to reduce the Pareto front to a single-quality indicator. HA is a measure of the quality of the performance of a two-objective optimisation algorithm. (Hypervolume is used for three or more objectives.) HA is commonly used to compare two Pareto fronts, either the true Pareto front and an approximated set (approximation to the true Pareto front), obtained from an optimisation algorithm, or two approximated sets obtained from two different optimisation algorithms. In the latter case, the optimisation algorithm that outperforms the other can be determined.

The HA indicator measures the area of the polygon between a Pareto front and a predetermined reference point. An example showing a polygon and a reference point is shown in Figure 3.7, where f_1 and f_2 must be minimised. It is clear that the HA indicator measures the relative spread of members of the Pareto optimal set and also the proximity of two Pareto optimal sets. A larger spread as well as solutions that are near-optimal will result in a larger HA. Additional material regarding HA and two techniques to determine the reference point can be found in Coello Coello *et al.* (2007) and Knowles *et al.* (2006).

In this study, the HA indicator is used to compare MOCBA, MOCBA_IZ and the EA. To have statistical confidence in the results of the comparison, it was necessary to create 50 pseudo-independent HA values via a trial by each algorithm. The Simio random streams, used to generate common random numbers, were varied at each stochastic point in the simulation models, for each of the 50 trials to ensure that a different solution per trial is produced. By default, Simio uses random stream zero. The HA values were then compared using standard statistical procedures. The algorithm with the largest significant HA can be deemed superior. A box-whisker plot with notches is also produced for the HAs to assist with the comparison.

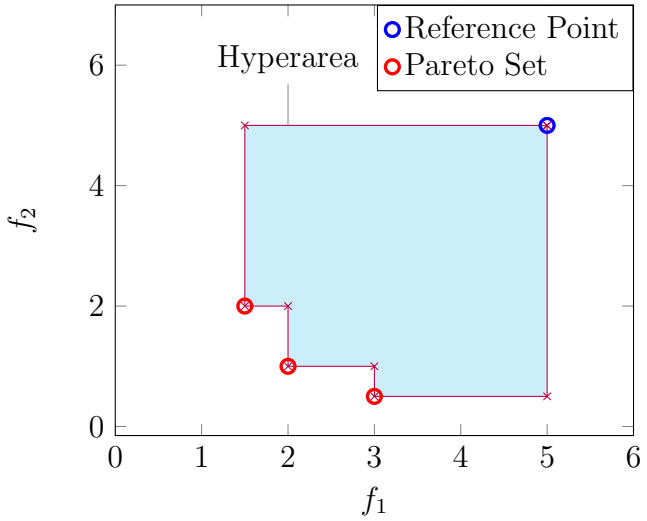


Figure 3.7: Example of a hyperarea and reference point.

3.4.5 Significance testing

The HA indicators are compared using standard statistical procedures, due to the stochastic nature of the simulation models. This will establish if an algorithm significantly outperforms the other in terms of HA. There are a number of methods that can be applied which include:

- the two-sample t-test,
- the Kruskal-Wallis test and
- the Mann-Whitney U-test.

The parametric two-sample t-test compares the means of two normal distributions, while the non-parametric Mann-Whitney U-test compares the median values of two distributions. The Kruskal-Wallis test is the equivalent of the Mann-Whitney U-test for three or more test sets (variables). It shows if any one of the test sets originate from a distribution that differs significantly from the distributions of the other test sets. The Mann-Whitney U-test was chosen (otherwise known as the Wilcoxon rank-sum test) since no assumption of the underlying output distributions is necessary and only two test sets exist. The null hypothesis of the test states that the data in the two test sets have equal medians, against the

alternative that a particular test set has larger values than the other (Knowles *et al.*, 2006). In this assessment, the right-tailed Mann-Whitney U-test was performed at a significance level of 5% on the HA output. The ranksum function in Matlab® was used. It returns the p-value of the test as well as a test decision, to reject or fail to reject the null hypothesis. The null hypothesis is rejected for any $p < \alpha$. Rejection of null hypotheses in favour of the alternative hypotheses is desirable.

3.5 Experimental results

This section contains the results from executing the experiments. A comparison is made between the MORS algorithms under the two test problems.

3.5.1 Summary and discussion of BAP results

To create 50 pseudo-independent trials for the BAP, the random number stream indices of Simio were set at the processing time of each of the five machines. These are the stochastic points in the model. These random number streams were chosen by randomly selecting values between zero and 10 (see Table 3.6). The results for the BAP follow.

3.5.1.1 BAP distribution of replications

Figures 3.8 and 3.9 illustrate how MOCBA and MOCBA_IZ allocate the simulation budget (for the BAP) when $R_T = 4\,000$ for Trial 1. As can be seen, replications are allocated differently for the two algorithms. Some of the scenarios receive no additional replications and their number of replications remains at the five initial replications run. In MOCBA the maximum number of replications run for a scenario, specifically scenario number 23, is 123 replications. Whereas, the maximum number of replications for a scenario in MOCBA_IZ is 58 (scenario 259). The smaller number of replications for MOCBA_IZ is due to the algorithm not spending as many replications differentiating between scenarios with close performances.

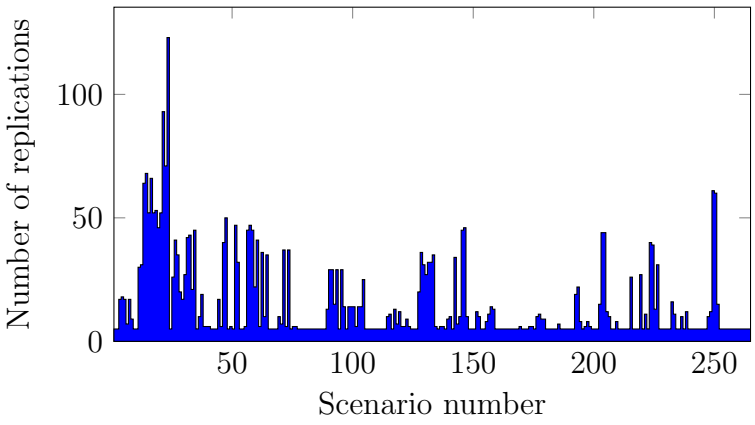


Figure 3.8: Replications distribution for MOCBA with the BAP.

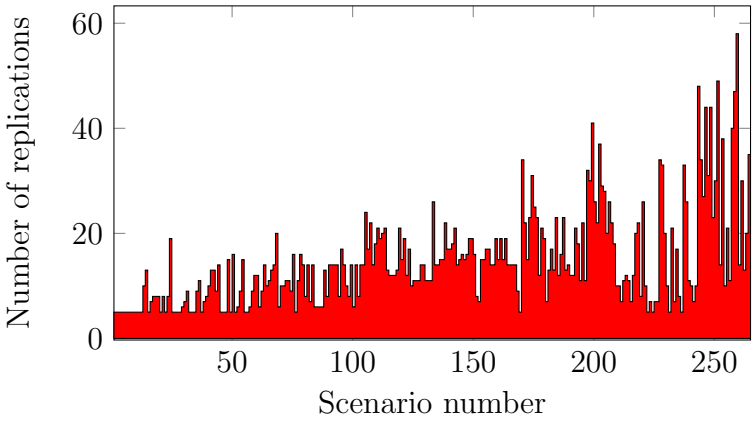


Figure 3.9: Replications distribution for MOCBA_IZ with the BAP.

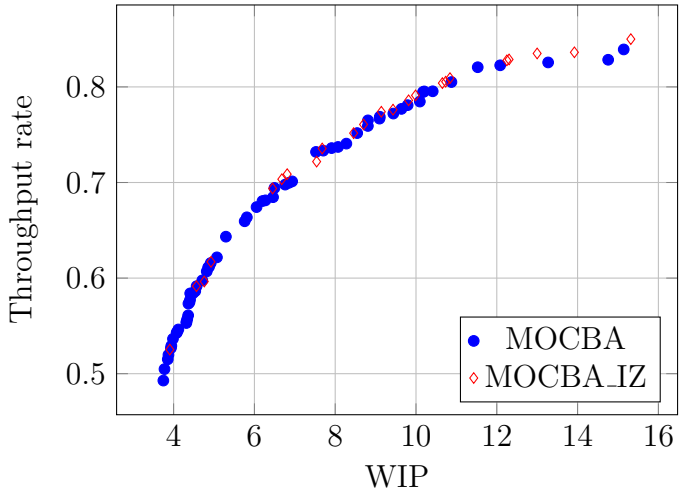


Figure 3.10: Pareto fronts achieved by the algorithms (Trial 1) for the BAP.

3.5.1.2 BAP Pareto fronts

The approximate fronts for the Trial 1 solution set of the BAP found by the MORS algorithms are shown in Figure 3.10. The Pareto front obtained from MOCBA contains 61 points while the front from MOCBA_IZ contains 23 points. The front from MOCBA_IZ contains fewer points due to the IZ ranking algorithm that incorporates an IZ. Besides the variation in the number of points in the two approximated Pareto sets, the fronts are dense in different areas. In addition, the fronts have similar spreads and are very close in proximity. From Figure 3.10 it cannot be concluded that one algorithm produces a better solution than the other.

3.5.1.3 BAP hyperarea

The HAs of the BAP comparison between MOCBA and MOCBA_IZ are shown in Table 3.6. It follows from the values in the table that the HAs for MOCBA are generally larger than for MOCBA_IZ. A supporting box plot is presented in Figure 3.11. It can also be seen from Figure 3.11 that the variability of the HAs for the algorithms are similar, in the BAP case.

In theory, the algorithm with the largest HA is classified as superior. However, a front that consists of more points will contain more area, compared to a front

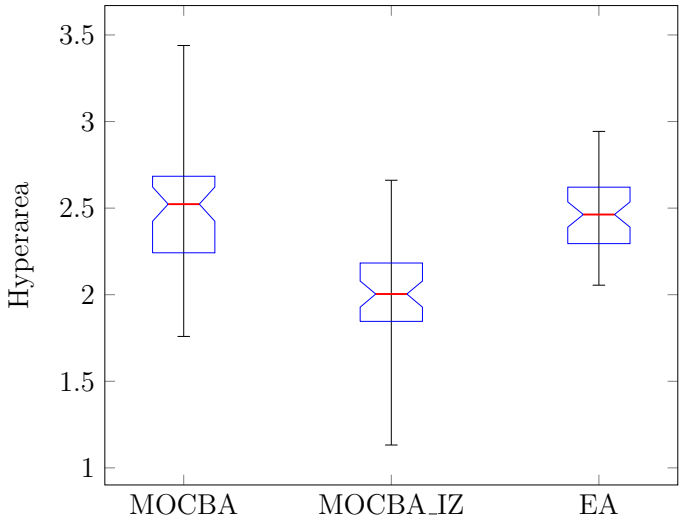


Figure 3.11: Box plot for the hyperarea comparison of the MOCBA algorithm, MOCBA_IZ algorithm and equal allocation using the BAP.

with less points, as some area is cut out. The MOCBA_IZ algorithm has an aggressive ranking approach which is why there are subsequently less scenarios in the Pareto front and the HA is lower, compared to MOCBA. Can the MOCBA algorithm truly be classified as superior for the BAP, considering the solution consists of more scenarios?

Next, the Mann-Whitney U test will determine if the MOCBA is significantly superior to the MOCBA_IZ.

3.5.1.4 Significance testing

The Mann-Whitney U-test was performed on the achieved HAs for the BAP. The formulation of the first hypothesis test is

$$\begin{aligned} H_0 &: m_{H_{\text{MOCBA}}} \leq m_{H_{\text{IZ}}} \\ H_1 &: m_{H_{\text{MOCBA}}} > m_{H_{\text{IZ}}} \end{aligned}$$

where $m_{H_{\text{MOCBA}}}$ is the mean of the HAs produced by the MOCBA algorithm, and $m_{H_{\text{IZ}}}$ is the mean of the HAs produced by the MOCBA_IZ algorithm. The outcome of the hypothesis test for the difference of the HAs for MOCBA and MOCBA_IZ is shown in Table 3.7. The p-value of zero indicates that the null

Table 3.6: Hyperareas for the MOCBA and MOCBA_IZ comparison using BAP. The Simio random number stream indices per trial are included.

Trial	Hyperarea MOCBA	Hyperarea MOCBA_IZ	Random stream index					Trial	Hyperarea MOCBA	Hyperarea MOCBA_IZ	Random stream index				
			M1	M2	M3	M4	M5				M1	M2	M3	M4	M5
1	2.855	2.501	6	1	9	4	8	26	2.478	2.373	8	5	2	3	2
2	3.121	2.089	7	8	10	3	6	27	2.481	1.873	4	6	4	5	9
3	1.927	2.155	10	10	9	10	2	28	2.808	1.753	6	4	4	5	4
4	2.551	2.338	3	1	5	9	7	29	2.513	1.843	7	8	6	5	9
5	1.970	1.838	9	10	8	5	7	30	2.569	2.456	8	9	3	8	8
6	2.951	2.350	5	5	1	7	10	31	2.241	2.302	4	3	9	6	1
7	2.223	2.366	3	2	3	6	8	32	2.144	1.474	9	1	8	6	6
8	2.545	2.026	1	5	6	1	3	33	2.724	2.445	4	5	4	5	3
9	2.679	2.142	6	7	4	8	2	34	2.085	1.828	9	4	6	6	2
10	2.582	1.763	10	6	8	5	8	35	2.216	1.977	1	5	5	1	8
11	2.761	1.509	3	2	2	3	6	36	3.065	1.242	2	6	3	2	4
12	2.056	1.433	9	3	8	5	3	37	2.602	2.124	10	4	9	2	10
13	2.344	1.900	9	8	10	4	7	38	2.450	2.029	4	2	8	9	10
14	2.573	1.981	8	2	9	5	4	39	2.055	2.105	2	5	2	4	10
15	2.554	1.940	10	7	2	1	9	40	2.671	2.599	1	9	5	2	2
16	3.439	1.292	7	3	5	5	9	41	2.650	1.898	5	5	2	1	1
17	2.091	1.973	8	5	5	7	2	42	1.759	2.193	9	5	5	1	5
18	2.061	1.227	6	7	5	2	9	43	3.012	2.131	7	8	3	9	5
19	3.371	2.146	8	7	2	10	10	44	2.525	1.132	4	10	9	8	6
20	2.521	1.877	6	2	6	1	8	45	2.659	2.661	7	5	9	1	7
21	2.329	2.124	1	6	4	5	6	46	2.488	1.854	3	10	1	6	4
22	2.331	1.978	4	5	2	1	1	47	2.025	1.939	9	1	4	6	8
23	2.247	2.326	3	8	3	2	6	48	2.828	2.217	8	10	7	7	7
24	2.686	2.151	5	4	5	3	7	49	2.713	2.064	3	8	6	4	3
25	2.243	1.948	5	4	8	6	3	50	2.389	1.303	6	2	5	4	7

Table 3.7: Outcome of the hypothesis test for the hyperarea indicator of the BAP: MOCBA and MOCBA_IZ.

	Hyperarea		
BAP test problem	p-value	t-stat	Outcome
MOCBA and MOCBA_IZ	0	6.0976	Reject H_0
MOCBA and EA	1	-8.6207	Do not Reject H_0

hypothesis of equal medians is rejected at a 5% significance level. Thus, the performance of the two algorithms are significantly different.

The HAs collected for the EA experiments were used to set up another hypothesis test between MOCBA and the EA rule. For the BAP, it was found that the results of MOCBA and EA are statistically the same, with regards to HA. The test-statistics can be found in Table 3.7.

Although the Pareto fronts in Figure 3.10 appear to be similar, this figure only shows the fronts for Trial 1, and from the Mann-Whitney U test on all the trials, it is known that the HAs of the algorithms are significantly different.

3.5.2 Summary and discussion of the inventory problem results

The inventory model has three stochastic points: the arrival rate, the demand, and the lead time. All these require randomly selected number stream indices for Simio, as shown in Table 3.8.

3.5.2.1 Inventory problem distribution of replications

Figure 3.12 and 3.13 show the number of replications allocated to each scenario by the MOCBA and MOCB_IZ allocation procedure, for the inventory problem. The figures represent the allocation for the first trial when $R_T = 3\,000$. As with the BAP, it can be seen that the algorithms allocate replications differently.

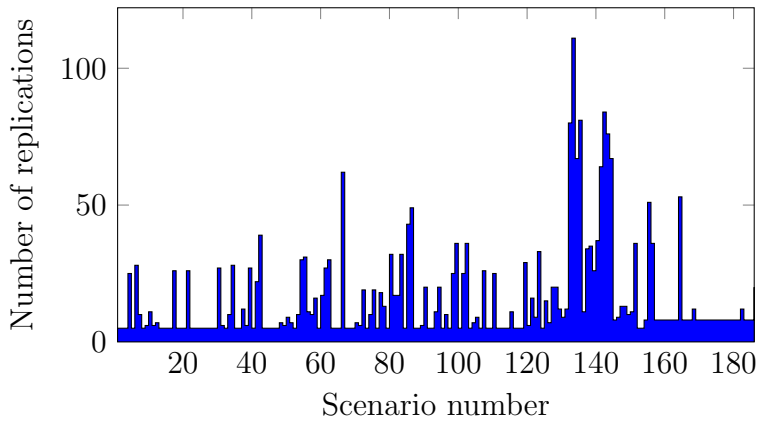


Figure 3.12: Replications distribution for MOCBA using the (s, S) inventory model.

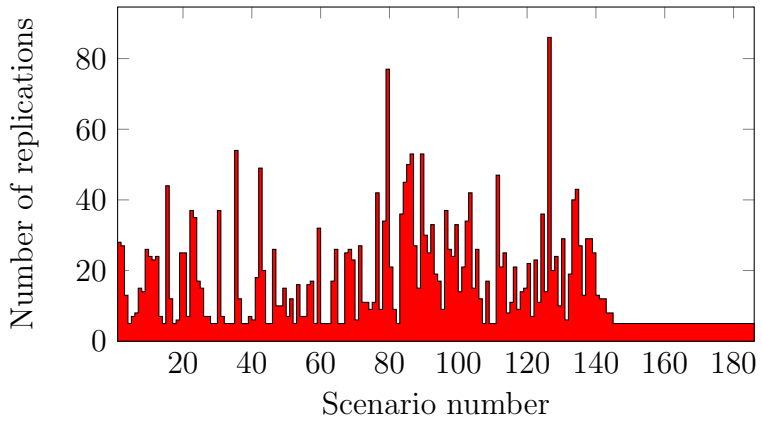


Figure 3.13: Replications distribution for MOCBA_IZ using the (s, S) inventory model.

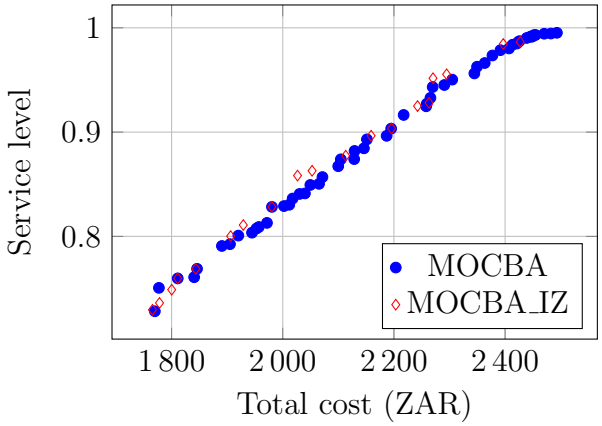


Figure 3.14: Pareto fronts achieved by the algorithms (Trial 1) for the inventory problem.

The higher scenarios (on the upper x-axis) consist of the solution space close to a 100% service level. These solutions are very close together with regard to the service level objective, which is why they do not receive additional replications in the MOCBA_IZ case.

3.5.2.2 Inventory problem Pareto fronts

Figure 3.14 shows the Pareto fronts achieved by the MOCBA and MOCBA_IZ algorithms for Trial 1. As before, the front from MOCBA_IZ consists of fewer points (19 points), than the front from MOCBA (55 points).

3.5.2.3 Inventory problem hyperarea

The HAs for the trials comparison of the MOCBA and MOCBA_IZ for the inventory model are shown in Table 3.8. Once again, the HAs of the MOCBA are generally larger than those of MOCBA_IZ.

A box plot of the HAs for the MORS algorithms is shown in Figure 3.15. In the inventory problem, the shape of the boxes are different. It is more likely that there will be variability in the HAs of MOCBA_IZ because its Pareto front consists of less points. The small variability in the HA of the EA could be due to a sufficient amount of simulation replications already allocated to the scenarios (16 replications), to obtain an accurate estimate of the performance measures.

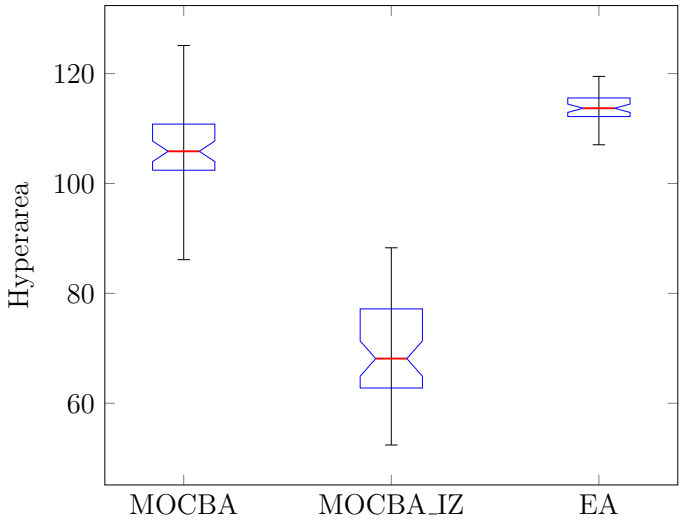


Figure 3.15: Box plot for the hyperarea comparison of the MOCBA algorithm, MOCBA_IZ algorithm and equal allocation using the (s, S) inventory model.

Does this mean that MOCBA and MOCBA_IZ need more simulation replications to get to solutions similar to EA?

3.5.2.4 Significance testing

The Mann-Whitney U-tests for the inventory problem were formulated in the same way as in the previous section. Hypothesis tests were performed for MOCBA and MOCBA_IZ as well as the EA rule and MOCBA. The results are shown in Table 3.9. It is concluded that the HA of MOCBA and MOCBA_IZ are significantly different. In addition, the HA of the EA rule and MOCBA are also significantly different.

From these experiments, it is concluded that the performance of the two MORS algorithms are significantly different. Is this only due to the large difference in the number of points in the Pareto sets from the two algorithms? In theory, the MOCBA procedure is superior (due to the larger HA), but a smaller Pareto front (MOCBA_IZ) is better for the decision maker as it makes decision making easier. On the other hand, when this method is combined with a search procedure, the Pareto front that is better for the search algorithm should be taken

Table 3.8: Hyperareas for the MOCBA and MOCBA_IZ comparison using the inventory problem. The Simio random number stream indices per trial are included.

Trial	Hyperarea MOCBA	Hyperarea MOCBA_IZ	Random stream index			Trial	Hyperarea MOCBA	Hyperarea MOCBA_IZ	Random stream index		
			Arrivals	Demand	Lead time				Arrivals	Demand	Lead time
1	106.647	88.296	1	8	2	26	106.188	75.740	4	6	6
2	109.303	61.114	7	9	1	27	116.217	64.735	4	2	8
3	104.018	84.479	6	6	1	28	99.830	66.520	5	1	1
4	108.154	64.073	6	6	7	29	111.468	63.651	3	5	9
5	102.137	73.836	8	8	9	30	110.068	85.702	9	9	10
6	103.231	74.415	2	1	6	31	109.486	84.613	9	3	5
7	104.020	52.404	4	8	2	32	108.229	83.006	2	8	5
8	118.930	78.379	8	3	3	33	103.064	63.166	7	6	1
9	110.947	73.102	5	6	1	34	103.380	66.620	5	5	3
10	102.708	59.460	8	7	1	35	101.130	60.175	1	7	10
11	112.896	59.135	1	9	9	36	98.546	67.011	3	8	3
12	118.019	64.843	9	10	10	37	114.345	69.256	2	2	3
13	109.318	64.959	5	4	2	38	95.573	85.322	7	1	9
14	111.904	77.576	5	3	10	39	99.375	52.993	5	1	10
15	125.107	65.688	8	10	9	40	114.102	72.829	5	6	6
16	113.107	74.560	4	5	3	41	107.431	75.147	8	2	8
17	104.553	59.620	3	6	5	42	103.185	86.720	1	1	10
18	102.315	72.319	6	3	5	43	89.658	57.583	7	7	6
19	99.291	62.678	6	2	8	44	105.356	66.936	4	10	5
20	114.794	80.989	9	5	2	45	115.099	84.613	9	3	5
21	110.398	62.730	3	3	4	46	106.068	62.916	6	9	3
22	110.149	73.586	3	2	9	47	100.500	70.983	8	9	7
23	105.159	78.174	6	10	2	48	105.642	61.333	6	2	7
24	103.582	75.981	2	10	6	49	101.255	80.023	8	2	9
25	100.798	53.361	8	5	8	50	86.130	61.755	7	5	6

Table 3.9: Outcome of the hypothesis test for the hyperarea indicator of the inventory problem: MOCBA and MOCBA_IZ.

	Hyperarea		
Inventory test problem	p-value	t-stat	Outcome
MOCBA and MOCBA_IZ	0	8.6001	Reject H_0
EA and MOCBA	0	5.9046	Reject H_0

into account. This is the topic of the following chapter using the MOO CEM as the search procedure.

For the MOO CEM, obtaining a larger set of elite solutions is better, as it guides the algorithm where to search for solutions (exploration).

3.6 Conclusion: Chapter 3

The aim of this chapter was to study MORS methods.

First the field of MOO was briefly introduced, followed by an introduction to R&S. Two main branches of R&S exist, namely OCBA methods and IZ methods. These concepts have recently been used to form MORS methods.

Two MORS methods were applied to two test problems. It was found that the two algorithms perform significantly differently, based on their HA indicators.

The next chapter advances to integrate a search algorithm with the MORS procedures. By doing this, the procedure will search for the best combinations of decision variables, and a set of possible scenarios are not required prior to experimentation.

Chapter 4

Incorporating multi-objective ranking and selection into the MOO CEM

This chapter focuses on integrating statistical ranking and selection (R&S) with the multi-objective optimisation method using the cross-entropy method (MOO CEM) algorithm. First, a brief overview is given on cases where the concepts of R&S and metaheuristics have already been combined. Next, the MOO CEM algorithm is presented, followed by a description of where the multi-objective ranking and selection (MORS) algorithms fit into the MOO CEM. Thereafter, the experiments and the results of integrating multi-objective optimal computing budget allocation (MOCBA) and MOCBA with an indifference-zone (MOCBA_IZ) with the MOO CEM are presented.

4.1 Introduction to integration of statistical selection with search mechanisms

In this chapter, optimisation problems are considered where the decision space is infinite, or finite yet very large. R&S methods can no longer be used alone, as they cannot efficiently simulate all the alternative scenarios. Search algorithms are implemented to intelligently explore the decision space for promising scenarios.

4.1 Introduction to integration of statistical selection with search mechanisms

Kim & Nelson (2007) suggest that R&S methods can be combined with search algorithms. In this case R&S will help the search algorithm find improved solutions with greater accuracy and efficiency (Kim & Nelson, 2006).

Search algorithms that require an elite set of solutions, either for reproduction of more promising solutions, or for an elite set kept for the next generation, provide an opportunity for R&S techniques to be incorporated (Juxin, 2012). The R&S process can be embedded in the search algorithm, where the solutions are ranked and the elite solution set is selected.

Another approach to simulation optimisation incorporating R&S, involves applying R&S in the last step of simulation optimisation. R&S procedures can be applied to solutions visited by the search algorithm, to provide a statistical guarantee that the solutions returned as best are at least the best of all the solutions evaluated (Kim & Nelson, 2006).

There are instances in literature where R&S, and search algorithms have been combined. The methods discussed below are not an exhaustive overview of all methods found in literature, but simply the ones that are most relevant to this study.

First, considering only one objective, Boesel *et al.* (2003) incorporate subset selection and indifference-zone ranking as the last stage of simulation optimisation, to select the best scenario. An example of where the CEM and R&S meet is where He *et al.* (2010) combine the CEM with the notion of optimal computing budget allocation, with the aim of improving efficiency. Numerical experiments show that the extended CEM has substantial computational efficiency gains over the CEM with equal allocation.

Secondly, a number of instances exist for multi-objective optimisation where search algorithms are combined with MOCBA. Lee *et al.* (2008) integrate multi-objective evolutionary algorithms with MOCBA and apply the combined method to a multi-objective aircraft spare parts allocation problem, to find a set of non-dominated solutions. Chew *et al.* (2009) integrate the nested partitions search method and MOCBA, and apply it to a differentiated service inventory problem.

A review of simulation optimisation methods that deal with randomness of the objective function(s), for example by using R&S is provided by Lee *et al.* (2006). They also present a general solution framework for integrating statistical selection

with search mechanisms. A flow chart showing how a general search procedure is integrated with MOCBA is given in Figure 4.1. In the figure, performance evaluation refers to obtaining values for the performance measures of the problem while, fitness evaluation refers to the process of selecting good solutions to the problem.

Juxin (2012) adapts two families of existing multi-objective evolutionary algorithms to include computing budget allocation. First, a general framework is provided for combining multi-objective genetic algorithms with MOCBA. The second family of evolutionary algorithms of interest are multi-objective estimation of distribution algorithms, which are combined with a MOCBA subset procedure.

In this study, the multi-objective optimisation method using the cross-entropy method by Bekker (2012) is the relevant search algorithm, which will be integrated with R&S. It is presented next.

4.2 Multi-objective optimisation with the cross-entropy method

The multi-objective optimisation using the cross-entropy method (MOO CEM) is presented in this section. The aim of the overview is to understand the method, specifically the ranking component, in order to integrate it with a MORS method. The reader is referred to Bekker & Aldrich (2010) and Bekker (2012) for a detailed explanation of the method.

Bekker & Aldrich (2010) develop an approach to multi-objective optimisation with the cross-entropy method. This recent endeavour shows promising results for a number of benchmark and practical multi-objective problems, as in cases, an approximation of the true Pareto front has been obtained with a relatively low number of simulations. The problems are summarised in Bekker (2012).

The foundation of the CEM was upheld while incorporating the multi-objective nature of the problem. To form a sample vector for decision variables, a truncated normal distribution is assigned for each decision variable x_i. It is truncated on the range of the x_i with mean μ_i and variance σ_i^2. The population of decision

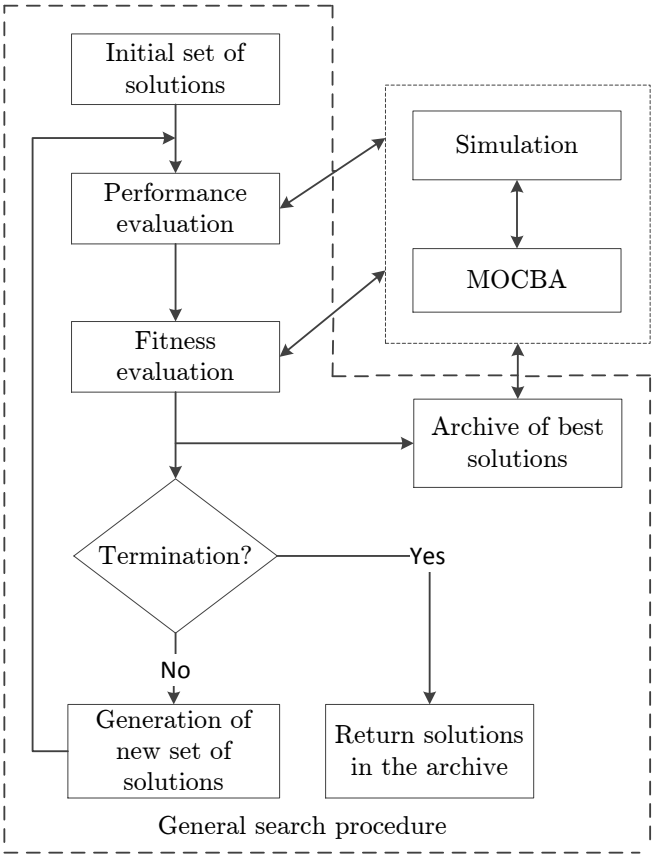


Figure 4.1: General framework: integration of MOCBA with search procedures (Lee *et al.*, 2006).

variables is populated with samples from the applicable truncated normal distribution for each decision variable. Next, the objective functions are evaluated using these decision variables.

In single-objective optimisation, the objective function values can easily be ranked to form an elite set, because only one objective function exists. This elite set consists of the $(1-\varrho)$-quantile of the population and is used to update the probability distribution of the decision variable. The $(1-\varrho)$-quantile cannot be estimated in MOO as the solution contains more than one objective value. This poses a challenge to expanding the CEM for use in MOO. To overcome this, Bekker & Aldrich (2010) introduce Pareto-ranking (Goldberg, 1989) to determine the best set of objective function values.

After evaluating the decision variables, the Pareto ranking algorithm (Algorithm 3) is applied to find the first three non-dominated Pareto fronts. They are stored in an elite vector, called *Elite*.

Using *Elite*, a histogram is constructed for each decision variable. The range of decision variable values is divided to form the bins of the histogram. The frequency of a bin is taken as the number of times a solution value is found in *Elite* that falls within a bin. An example of a histogram with seven bins is shown in Figure 4.2. The first bin begins at the lower limit of the range of the decision variable and ends at the minimum value of the decision variable found in the elite. The frequency in this bin is zero. Similarly, the last bin begins at the maximum value found in the elite for the decision variable and ends at the upper limit value of the decision variable. Due to rounding errors, the last bin might not have a frequency of zero. The five bins that remain are assigned equal sizes.

For the next iteration of the algorithm, the population of new decision variables is created proportionally to the bin frequencies, for each decision variable. For example, if 15% of the decision variable values in *Elite* fall in the second bin, then in the next iteration, 15% of the variable values of the population will be selected from the same region. For each bin, the decision variables are sampled from a truncated normal distribution with parameters $\mu_i = \text{LB bin} + U(0, \text{UB bin} - \text{LB bin})$ and $\sigma_i^2 = \text{UB bin} - \text{LB bin}$. LB and UB refer to the upper and lower bound, respectively.

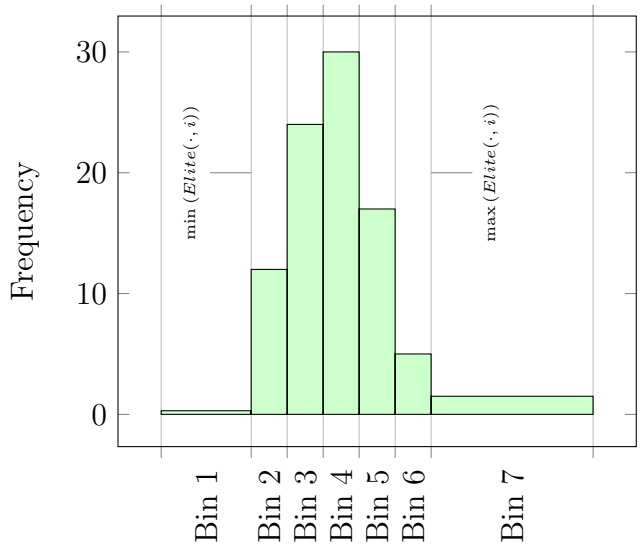


Figure 4.2: Example of a histogram for the decision variable x_i.

The concept of inverted histograms is used in the algorithm to maintain population diversity and avoid early convergence on a local optimum. Histograms are inverted with a probability p_h (typically $p_h = 0.1$ to $p_h = 0.3$). Inversion entails subtracting the frequencies of the bins from the maximum frequency occurring in the histogram (over all the bins). For example, if 15% of the decision variable values in *Elite* fall in the second bin, and the histogram is inverted, then in the next iteration, 85% of the variable values will fall within the same region. An inverted histogram is shown in Figure 4.3.

The above process is repeated a number of times to ensure a good balance between exploration and exploitation. At each iteration, the solutions are ranked using a threshold of $\rho_E = 2$ and solutions with a ranking value of zero to two are added to the current *Elite*. In this way, *Elite* is built up from the best solutions of the last few generations. This is known as elitism by which the best solutions found in the generation are carried over to the next generation to avoid losing these solutions.

The parameter vectors (μ_i, σ_i) are smoothed using the decision variables values of *Elite*, in the same way as in the CEM in (2.9). After a certain number of generations or if σ_i of each decision variable has decreased below a common

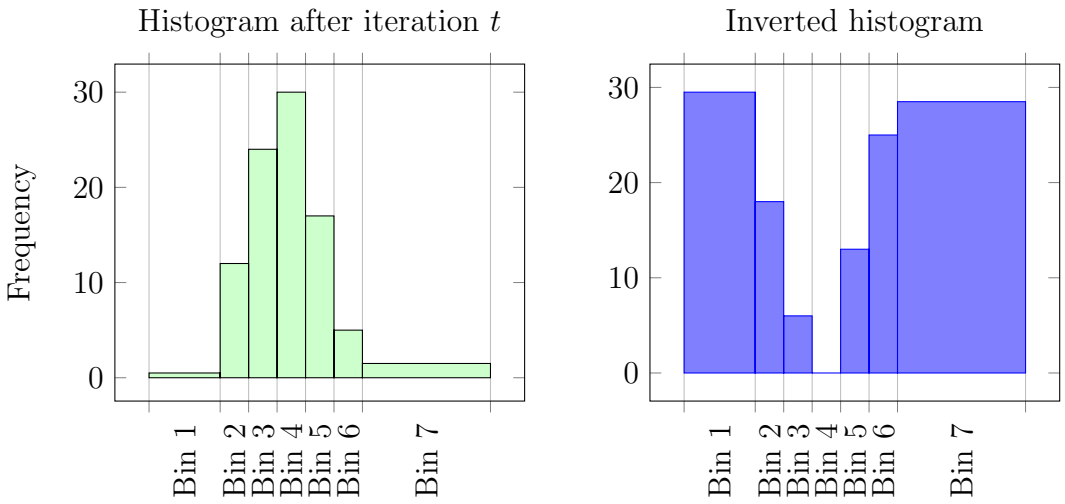


Figure 4.3: The effect of adjusting histogram frequencies for the decision variable x_i.

threshold ϵ_c, *Elite* is once again ranked ($\rho_E = 1$) and the first two non-dominated fronts are retained.

The algorithm reiterates this process until a specified number of loops have been performed. When the algorithm terminates, *Elite* is ranked one last time using a threshold of $\rho_E = 0$ to determine the non-dominated solutions. The basic MOO CEM algorithm is presented in Algorithm 8.

4.3 Integrating MOCBA and MOCBA_IZ into the MOO CEM

The MOO CEM algorithm has three locations where ranking takes place, using the Pareto ranking algorithm by (Goldberg, 1989). At each location, the ranking is performed at a different threshold, according to the requirements of the algorithm. For example, the first layer of ranking is lenient and many good solutions are retained, while the third layer of ranking is strict and only the non-dominated solutions are retained. This provides many options for introducing statistical R&S into the algorithm.

In this study, the MORS algorithms were inserted into the MOO CEM where

Algorithm 8 MOO CEM Algorithm (Bekker, 2012)

Create initial variable vectors and compute the initial objective values.
Rank the initial population to find *Elite* using the Pareto ranking of Algorithm 3 with $\rho_E = 2$.
while stopping criteria not met,
    for each variable i **do**
        Create histogram i using *Elite*.
        if $rand(0,1) \leq p_h$ **then**
            Invert histogram.
        end if
        for each bin in histogram i **do**
            Create $\lfloor \frac{\text{Frequency of bin} \times \text{Population size}}{\text{Size of Elite}} \rfloor$ new decision variable vectors using a truncated normal distribution.
        end for
    end for
    Compute the objective function values of the new decision variable vectors.
    Rank the objective function values with $\rho_E = 2$ and add the decision variables vectors with a ranking value of zero to two to *Elite*.
    Smooth the parameter vectors for all decision variables.
    if all $\sigma_i < \epsilon_c$ or more than the allowable number of evaluations has been performed **then**,
        Rank the elite vector *Elite* with $\rho_E = 1$.
    end if
end while
Rank the elite vector *Elite* with $\rho_E = 0$ to obtain the final elite vector.

previously, the first stage of ranking took place. This is done with a Pareto ranking threshold of one so that the first two fronts are appended to *Elite*. In this way an archive of good solutions can still be built-up to guide the search algorithm. The second place where solutions were previously ranked is not used any more. When the MOO CEM terminates, *Elite* is ranked using the Pareto ranking method (Algorithm 3), with $p_E = 0$ to determine the Pareto set.

4.4 Experimental design

MOCBA and MOCBA_IZ were integrated into the MOO CEM, which currently does not utilize a R&S approach to determine the non-dominated scenarios. The MOO CEM algorithm including MOCBA and MOCBA_IZ were implemented in Matlab®. The buffer allocation problem (BAP) and inventory model, in Simio, were again used as test problems, albeit with slight modifications.

4.4.1 Additions to the buffer allocation problem

The BAP previously presented as a test problem is used again in this section, with some input data modifications to create two cases. Experiments are performed on the same cases in Bekker (2012), using the MOO CEM as an optimiser. The instances of the BAP considered are:

1. BAP1: The same as the BAP model experimented on in the previous chapter with exponential processing and repair times. The capacity of the buffer sizes for the machines are limited by a maximum value given in Table 4.1, as suggested by Bekker (2012).

2. BAP2: Five machines are added to this manufacturing line. The processing times are exponential with rates $\mu_1 = 8$, $\mu_2 = 8$, $\mu_3 = 11$, $\mu_4 = 14$, $\mu_5 = 14$, $\mu_6 = 11$, $\mu_7 = 8$, $\mu_8 = 8$, $\mu_9 = 6$, $\mu_{10} = 6$. The failure rates are $\lambda_i = 1/19$ and the repair rates are $r_i = 0.5$ for all machines. The maximum buffer size for all machines is 200.

Table 4.1: Model buffer sizes for BAP1.

	Buffer sizes allowed
Machine	BAP1
2	35
3	30
4	25
5	25

The simulation time of the model was extended to 11 000 simulation time units long, with the first 1 000 simulation time units acting as the warm-up period.

For the BAP instances described above, Bekker (2012) modified the work-in-progress (WIP) performance measure for optimisation. Instead of estimating the mean WIP level, the actual number of units in the system due to observed WIP as it persists over time, is estimated. As the WIP fluctuates over time, the time duration of the WIP at each level is measured, and after a finite amount of time, the p_q-th percentile of these time-based levels is determined. In this way, the intensity of each WIP level is considered.

The benefit of this is described by means of an example. Suppose only the WIP level is observed and a few extreme high levels occur, for a short period. This will influence the WIP estimate and more buffer spaces will be allocated. Realistically, these spaces will not be used often. The time-based percentile objective will ensure buffer size extremities that exist for a short time, will not have a significant effect on WIP.

A time persistent graph of the new objective of BAP is illustrated in Figure 4.4. Suppose a decision maker is interested in the 95-th percentile of the WIP intensity and their WIP over time is as depicted in Figure 4.4. Four buffer slots will be recommended to the decision maker, as 95% of the observed WIP falls below $W_P(\mathbf{x}) = 4$. The details of the calculation can be found in Bekker (2012).

4.4.2 A modified inventory problem

A few revisions were made to the inventory model of Subsection 3.4.1.2, in order to make it comparable to the model presented in Bekker (2012). By doing this,

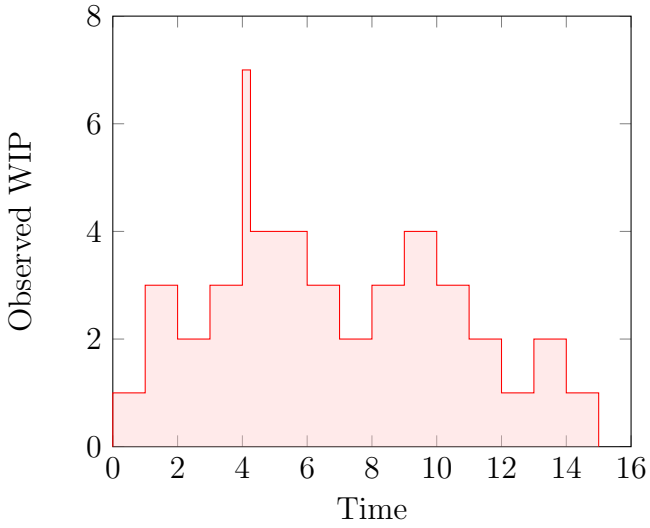


Figure 4.4: A graph illustrating WIP intensities over time.

the results of the MOO CEM including a MORS algorithm can be compared to the results of the MOO CEM for the inventory problem in Bekker (2012).

The alterations to the inventory model are as follows: Customers arrive to purchase goods according to a Poisson process, such that interarrival times are exponentially distributed with a mean of 0.5 hours. The discrete demand of a customer is distributed $\lfloor 20 \cdot beta(2,1) \rfloor$. When the inventory in the system reaches the reorder level s, then the reorder quantity S is ordered. The delay between the notification and the delivery of S is $U(1,3)$ hours. The cost associated with each order is ZAR100 and the holding cost is taken as ZAR10/unit/hour. The inventory system simulated operates for 2 000 minutes following a warm-up period of 100 minutes. The initial inventory in the system was taken as 100 units.

Note that the inventory model from Bekker (2012) is implemented in the simulation package Arena.

4.4.3 Settings for the algorithms

The experimental setup for the experiments conducted for this part of the research is given below.

4.4.3.1 Setting for the buffer allocation problem

The MORS algorithms parameters were chosen as follows:

- The MOCBA parameters were: $R_T = 600$, $n_0 = 5$, $\tau = 20$ and $\Delta = 100$.
- For MOCBA_IZ, in addition to the above, the IZ are $T_R(\mathbf{x}) = 0.005$ and $W_P(\mathbf{x}) = 0.1$.

The values set for the MOO CEM were: $N = 20$, $p_q = 0.95$, $p_h = 0.2$, $\delta = 10^{-2}$ and $\alpha = 0.7$.

When experimenting with the BAP, Bekker (2012) imposed upper bounds on the size of the buffers (see Section 4.4.1). The probability matrix of the CEM can become large if the quantity of buffers is large, and there is a large number of slots allowed per buffer. For this reason, the experiments in Bekker (2012) sampled the population of decision variables from truncated Poisson distributions on the range of the buffer sizes.

Experiments performed on the MOO CEM including MORS were not implemented using truncated Poisson distributions but with the original truncated normal distributions. It was assumed that this approach would still be valid.

4.4.3.2 Settings for the inventory problem

- The parameters for the MOCBA were: $R_T = 1\,000$, $n_0 = 5$, $\tau = 20$ and $\Delta = 100$.
- For MOCBA_IZ, in addition to the above, the IZ for cost is 5 and for service level is 0.01.

The settings for the MOO CEM algorithm were taken as $N = 30$, $\epsilon_c = 2.5$, $p_h = 0.2$, $\alpha = 0.7$ and the number of main loops was 10. When the original MOO CEM was applied to the inventory model, each (s, S) combination was evaluated using five simulation replications.

4.5 Experimental results

The results for the modified BAP and inventory problems are presented and discussed in this section.

4.5.1 Buffer allocation problem

The Pareto fronts obtained from experimentation with BAP1 are shown in Figure 4.5. The Pareto fronts from the MOO CEM including MORS and without MORS are in very close proximity. In this case, including MORS in the MOO CEM did not have an impact on the Pareto front found by the algorithm.

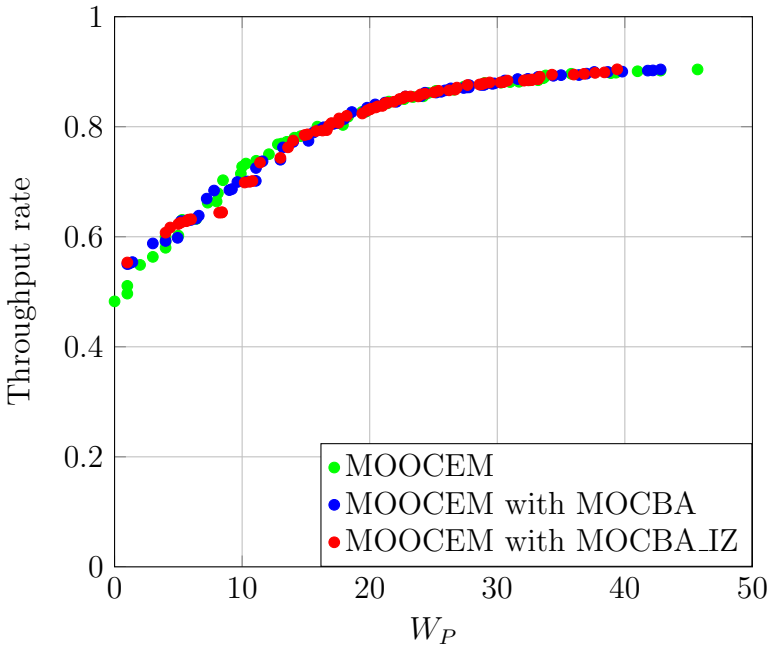


Figure 4.5: Pareto fronts for BAP1.

For BAP2, the resulting Pareto fronts from adding MORS to the MOO CEM, together with the front obtained using the original MOO CEM, are shown in Figure 4.6. As can be seen in the figure, all the Pareto fronts obtained have a similar shape. The MOO CEM Pareto front has a better spread for lower throughput rate and WIP values, while the algorithm including MORS has a better spread for higher objective function values. The Pareto front found by

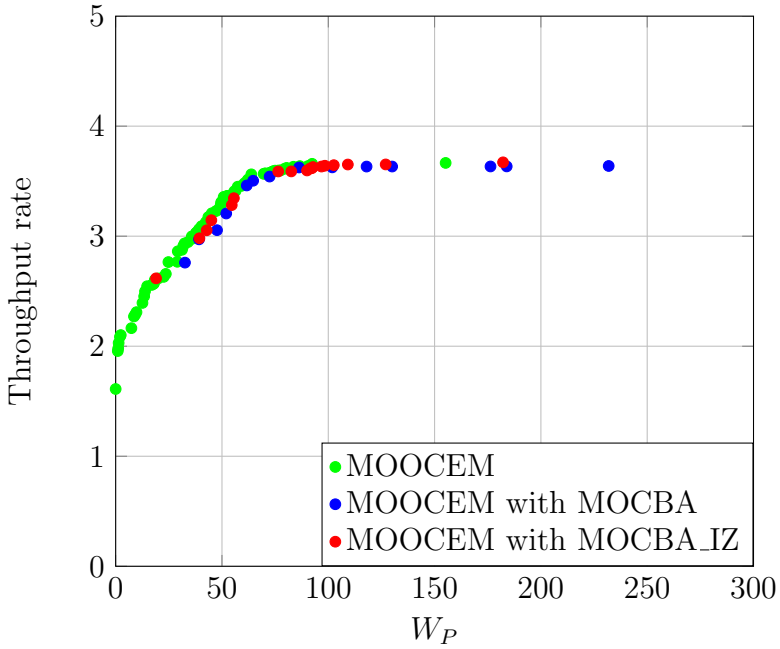


Figure 4.6: Pareto fronts for BAP2.

MOO CEM contains more scenarios and is slightly improved, when compared to the front found by the algorithm including MORS.

4.5.2 Inventory problem

The Pareto fronts obtained from experimentation are shown in Figure 4.7 for the inventory problem. Near 100% service level, all algorithms found similar solutions. However, in the case that the decision maker is aiming for a slightly lower service level, the MOO CEM found better solutions without MORS.

The MOO CEM with MOCBA algorithm included a few scenarios in the approximated Pareto set that have a service level of 100%. Some of the scenarios have a very large total inventory cost (ZAR7910). These scenarios are redundant because if the decision maker seeks 100% service level, they will favour the smallest cost.

The Pareto fronts of MOO CEM with MOCBA and MOO CEM with MOCBA_IZ are fairly well spread, although no solutions can be found with a service level of between 70% and 85% for MOO CEM with MOCBA. Similarly for MOO CEM

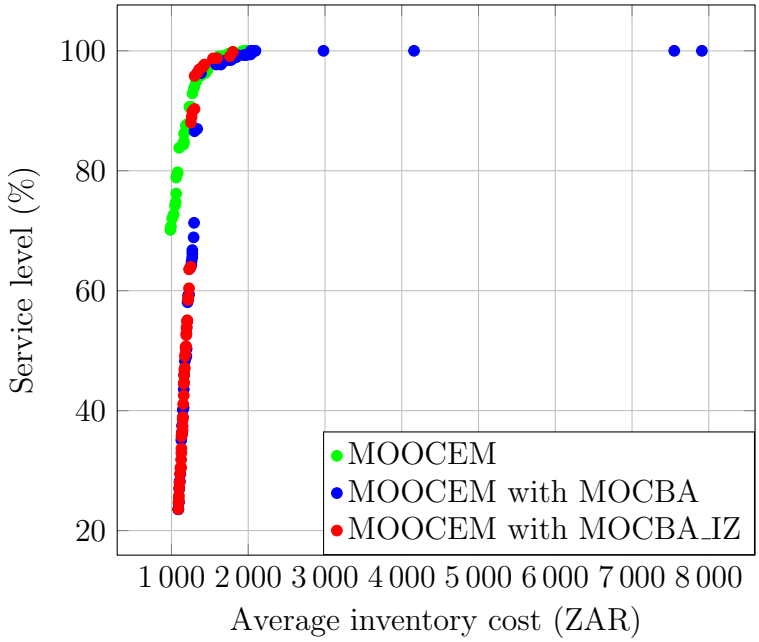


Figure 4.7: Pareto fronts for the (s, S) inventory problem.

with MOCBA_IZ, no solutions can be found with a service level of between 64% and 88%. In this case, the decision maker would not be able to select a solution with a service level of 80% for example. Even though the Pareto fronts from using MORS with MOO CEM include many points with a low service level, it is unlikely that a decision maker would select a service level below at least 60%.

From experimentation, it is concluded that the algorithm incorporating MORS into the MOO CEM has various effects on the non-dominated set of solutions obtained. Prior to experimenting with different problems, the decision maker has no way of knowing what result will be true.

It must be noted that, to comprehend the stochastic nature of the problems to be solved, and to be more confident that solutions in the Pareto set belong there, R&S is necessary. However, this does not necessarily mean that an improved Pareto set will result.

The common trade-off in society between time and quality once again emerges: Should one aim for the least simulation run-time at the expense of solution quality,

or is sufficient value gained by exploiting solutions, to justify the additional runtime?

4.6 Conclusion: Chapter 4

This chapter presents the first attempt at integrating statistical R&S with the MOO CEM algorithm. A few instances in literature where R&S and search algorithms have been integrated were briefly discussed. After this, the search algorithm specific to this study, the MOO CEM, is presented. Next, details are provided on how the MOO CEM and the MORS algorithms were integrated. This is followed by the experimental procedure taken and the results from applying the extended MOO CEM to the BAP and inventory problem.

Incorporating MORS into the MOO CEM algorithm displayed mixed results for the problems. Additional research on different problems is necessary to further investigate the proposed combined algorithm.

The next chapter presents a summary of the research completed, primary findings, recommendations for possible future research projects, the value of the study to the research field and personal skills acquired.

Chapter 5

Research summary and conclusions

The research project conducted was presented in the previous chapters. This chapter serves as a summary of the findings of the study and suggestions for future work.

5.1 Project summary

The purpose of this research project was to investigate ranking and selection (R&S) methods for the multi-objective optimisation method using the cross-entropy method (MOO CEM). This was introduced in **Chapter 1**. To achieve the aim, a methodology was followed through the research process.

In **Chapter 2**, the cross-entropy method was presented with a focus on discrete optimisation. After this, the water management optimisation study was presented. Although there is a great deal of future work on the combined optimisation - JAMS/J2000-NaCl model, and there was little support from stakeholders, the aim of understanding the cross-entropy method by applying it to a practical problem was achieved.

Chapter 3 considers multi-objective ranking and selection (MORS) procedures and describes, in detail, three methods that are of interest in this study. The multi-objective optimal computing budget allocation (MOCBA) algorithm and the multi-objective optimal computing budget allocation with indifference-zone

(MOCBA_IZ) algorithm were subsequently applied to two stochastic problems that are common to literature. These are the buffer allocation problem and an inventory problem. The problems were implemented in Simio and the algorithms were coded in Matlab®. The simulation model and optimisation model were integrated using the API of the simulation package.

The performance of the algorithms was measured using the hyperarea indicator and the Mann-Whitney U-test. In both problems, it was concluded that MOCBA significantly outperformed MOCBA_IZ. However, this may be due to the Pareto fronts obtained from using MOCBA_IZ consisting of fewer points, in which case another indicator might be more useful. For the inventory model, applying the equal allocation rule resulted in a significantly better performance, with regards to hyperarea, than applying MOCBA. In this case, it was simple to choose the equal number of simulation replications to execute, because the experimentation for MOCBA and MOCBA_IZ had already been performed. However, this would not be possible without prior information.

The last part of the research is conducted in **Chapter 4**, where statistical R&S is incorporated into the MOO CEM. This was the main aim of this study. Both the MOCBA and MOCBA_IZ algorithm were incorporated into the MOO CEM, to replace the previous ranking method.

Results from optimising a buffer allocation problem and an inventory problem, using the MOO CEM were taken from Bekker (2012). Experiments using the MOO CEM with R&S were performed on the same simulation models, so the results could be fairly compared. Results for the buffer allocation problem showed that the MOO CEM with R&S obtained similar Pareto sets to the Pareto front from the MOO CEM. In the case of the inventory model, the MOO CEM without R&S produced better results.

It was concluded that the application of the extended MOO CEM procedures was achieved with reasonable success and the overall aim of the research was accomplished.

5.2 Suggestions for further research

Suggestions for future work specific to the water optimisation problem is provided in Section 2.8. Possible future research related to the rest of the study includes:

1. Coding the two-stage-Pareto-set-selection procedure and applying it to stochastic multi-objective optimisation problems.
2. Applying the MORS methods to more problems and comparing their performance.
3. In order to provide any statistical confidence on the extended MOO CEM algorithm's performance, a number of pseudo-independent trials need to be performed by repeating the experiments and changing the random streams in Simio for each trial.
4. Applying the MOO CEM with R&S to more problems to investigate the performance of the combined algorithm.
5. Further analysing where in the MOO CEM MORS should be incorporated.
6. Comparing the MOO CEM with R&S to other metaheuristics.
7. The aim when proposing the MOO CEM in Bekker (2012) was to develop an efficient algorithm that provides near-optimised results with minimal evaluations of the objective function values. Incorporating R&S into the MOO CEM adds computational time to the algorithm. Research could be performed to determine the effect incorporating R&S with the MOO CEM has on the efficiency of the algorithm.

5.3 Value of the study

The work completed in this study has value by making contributions to the research field. The three main contributions are:

1. Providing a thorough literature overview of MORS methods. Only methods that do not require preference information were of interest.

2. Applying the fairly new MOCBA algorithm and the MOCBA_IZ algorithm to two new problems — the buffer allocation problem and an (s, S) inventory problem.

3. Investigating the effect and the outcome of integrating statistical R&S into the MOO CEM.

5.4 Skills acquired

A Masters student has to prove mastering some aspects of a research field. Through completing the research project, the researcher learnt a number of skills which include the following:

1. Mastering the use of software packages LaTeX, Matlab® and Simio.

2. Obtaining a basic proficiency of using `C#`.

3. Gaining a richer understanding on the topics of

   - simulation optimisation,
   - metaheuristics (CEM and MOO CEM),
   - multi-objective optimisation, and
   - statistical R&S.

4. Mastering how to apply the cross-entropy method for optimisation and using the MOO CEM.

5. Mastering and implementing the MOCBA and the MOCBA_IZ algorithm to various problems.

6. These days researchers have an abundance of software tools available, and it is often required to harness a suite of tools to obtain optimisation, or even near-optimisation. In addition, in the case of simulation optimisation, these tools often have to be integrated to provide information for one another. In this study, the tools connected were the JAMS/J2000-NaCl model and the CEM optimisation algorithm in Matlab®, and Simio and Matlab®.

7. The researcher learnt important lessons about working with stakeholders of a real-world project (in the case of the water management study).

References

Alrefaei, M.H. & Alawneh, A.J. (2004). Selecting the best stochastic system for large scale problems in DEDS. *Mathematics and computers in simulation*, **64**, 237–245. 37

Bashyam, S. & Fu, M. (1998). Optimization of (s, S) inventory systems with random lead times and a service level constraint. *Management Science*, **44**, 243–256. 71

Bechhofer, R.E. (1954). A single-sample multiple decision procedure for ranking means of normal populations with known variances. *The Annals of Mathematical Statistics*, 16–39. 36

Bekker, J. (2012). *Applying the cross-entropy method in multi-objective optimisation of dynamic stochastic systems*. Ph.D. thesis, Stellenbosch University. iii, v, 3, 5, 70, 91, 96, 97, 98, 99, 100, 106, 107

Bekker, J. & Aldrich, C. (2010). The cross-entropy method in multi-objective optimisation: An assessment. *European Journal of Operational Research*, **211**, 112–121. iii, v, 3, 29, 91, 93

Boesel, J., Nelson, B.L. & Kim, S.H. (2003). Using ranking and selection to clean up after simulation optimization. *Operations Research*, **51**, 814–825. 90

Branke, J., Chick, S.E. & Schmidt, C. (2007). Selecting a selection procedure. *Management Science*, **53**, 1916–1932. 35, 75

Bugan, R.D.H. (2008). *Hydrosalinity fluxes in a small scale catchment of the Berg River (Western Cape)*. Master's thesis, University of the Western Cape, South Africa. 11

Bugan, R.D.H. (2014). *Modeling and regulating hydrosalinity dynamics in the Sandspruit river catchment (Western Cape)*. Ph.D. thesis, Stellenbosch University. 14

Bugan, R.D.H., Jovanovic, N. & de Clercq, W.P. (2012a). The water balance of a seasonal stream in the semi-arid Western Cape (South Africa). *Water SA*, **38**, 201–212. 12, 13, 14, 20

Bugan, R.D.H., Jovanovic, N., Fink, M., Steudel, T. & Pfenning, B. (2012b). Sandspruit Hydrosalinty Modelling. Tech. Rep. No. K5/1846, Deliverable 27, Water Research Commission, Pretoria. 11, 12, 13, 15

Bugan, R.D.H., Fink, M., Jovanovic, N., de Clercq, W.P., Helmschrot, J., Steudel, T., Pfenning, B. & Fischer, C. (2013). Dryland Salinity Management: Land Use Change Scenarios. Tech. Rep. K5/2063, Deliverable 5, Council for Scientific and Industrial Research, Stellenbosch. 15, 16, 19, 25

Butler, J., Morrice, D.J. & Mullarkey, P.W. (2001). A multiple attribute utility theory approach to ranking and selection. *Management Science*, **47**, 800–816. 39

Chapman, V. (1966). Vegetation and Salinity. In *Salinity and Aridity*, Monographiae Biologicae, 23–42, Springer. 14

Chen, C.h. & Lee, L.H. (2010). *Stochastic simulation optimization: An optimal computing budget allocation*, vol. 1. World Scientific. 41, 45, 46, 47, 50, 51, 52, 53, 54, 55, 56, A-1

Chen, C.H., Chen, H.C. & Dai, L. (1996). A gradient approach for smartly allocating computing budget for discrete event simulation. In *Proceedings of the 28th Winter Simulation Conference*, 398–405, IEEE Computer Society. 2, 37, 38

Chen, C.H., Lin, J., Yücesan, E. & Chick, S.E. (2000a). Simulation budget allocation for further enhancing the efficiency of ordinal optimization. *Discrete Event Dynamic Systems*, **10**, 251–270. 37

Chen, C.H., Donohue, K., Yücesan, E. & Lin, J. (2003). Optimal computing budget allocation for Monte Carlo simulation with application to product design. *Simulation Modelling Practice and Theory*, **11**, 57–74. 37

Chen, E.J. (2007). Indifference-zone subset selection procedures: using sample means to improve efficiency. In *Proceedings of the 39th Winter Simulation Conference: 40 years. The best is yet to come*, 535–543, IEEE Press. 64

Chen, E.J. & Kelton, W.D. (2005). Sequential selection procedures: Using sample means to improve efficiency. *European Journal of Operational Research*, **166**, 133–153. 37

Chen, E.J. & Lee, L.H. (2009). A multi-objective selection procedure of determining a pareto set. *Computers & Operations Research*, **36**, 1872–1879. 40, 64, 65, 67

Chen, E.J. & Li, M. (2010). A new approach to estimate the critical constant of selection procedures. *Advances in Decision Sciences*, **2010**. 51, 64

Chen, E.J., Chen, C.H. & Kelton, W.D. (2004). Optimal computing budget allocation of indifference-zone-selection procedures. *Submitted*. 38

Chen, H.C., Dai, L., Chen, C.H. & Yücesan, E. (1997). New development of optimal computing budget allocation for discrete event simulation. In *Proceedings of the 29th Winter Simulation Conference*, 334–341, IEEE Computer Society. 37

Chen, H.C., Chen, C.H. & Yücesan, E. (2000b). Computing efforts allocation for ordinal optimization and discrete event simulation. *Automatic Control, IEEE Transactions on*, **45**, 960–964. 37

Chew, E.P., Hay Lee, L., Teng, S. & Hwee Koh, C. (2009). Differentiated service inventory optimization using nested partitions and MOCBA. *Computers & Operations Research*, **36**, 1703–1710. 90

Chick, S.E. (1997). Selecting the best system: A decision-theoretic approach. In *Proceedings of the 29th Winter Simulation Conference*, 326–333, IEEE Computer Society. 35

Chick, S.E., Inoue, K., Inoue, K. & Inoue, K. (2001). New two-stage and sequential procedures for selecting the best simulated system. *Operations Research*, **49**, 732–743. 36

Coello Coello, C.A. (2009). Evolutionary multi-objective optimization: some current research trends and topics that remain to be explored. *Frontiers of Computer Science in China*, **3**, 18–30. 31

Coello Coello, C.A., Lamont, G.B. & Van Veldhuizen, D.A. (2007). *Evolutionary Algorithms for Solving Multi-Objective Problems*. Springer, 2nd edn. 31, 76

Cruz, F., Duarte, A. & Van Woensel, T. (2008). Buffer allocation in general single-server queueing networks. *Computers & Operations Research*, **35**, 3581–3598, Part Special Issue: Topics in Real-time Supply Chain Management. 70

Cruz, F., Van Woensel, T. & Smith, J.M. (2010). Buffer and throughput trade-offs in M/G/1/K queueing networks: A bi-criteria approach. *International Journal of Production Economics*, **125**, 224–234. 68

de Clercq, W.P., Jovanovic, N. & Fey, M. (2010). Land use impacts on salinity in Berg River water Research on Berg River water management. Tech. Rep. No. 1503, Water Research Commission, Pretoria. 12, 14

de Clercq, W.P., Bugan, R.D.H. & Jovanovic, N. (2013). Implementation of salinity and water management tools for the Berg and Breede catchments in the Western Cape. Tech. Rep. No. K5/2063, Water Research Commission, Pretoria. 14, 21

De Weck, O.L. (2004). Multiobjective optimization: History and promise. In *Invited Keynote Paper, GL2-2, The Third China-Japan-Korea Joint Symposium on Optimization of Structural and Mechanical Systems, Kanazawa, Japan*, vol. 2. 34

Department of Water Affairs and Forestry, South Africa (2004). Berg water management area: Internal strategic perspective. DWAF Report No. PWMA 19/000/00/0304, Prepared by Ninham Shand (Pty) ltd in association with Jakoet and Associates, Umvoto Africa and Tlou and Matji, on behalf of the Directorate: National Water Resource Planning. 12

Dudewicz, E.J. & Dalal, S.R. (1975). Allocation of observations in ranking and selection with unequal variances. *Sankhyā: The Indian Journal of Statistics, Series B*, 28–78. 36

Dudewicz, E.J. & Taneja, V.S. (1978). Multivariate ranking and selection without reduction to a univariate problem. In *Proceedings of the 10th Winter Simulation Conference-Volume 1*, 207–210, IEEE Press.

Fey, M. & de Clercq, W.P. (2004). Dryland salinity impacts on Western Cape rivers. Report no. 1342/1/04, Water Research Commission, Pretoria. 11, 12

Fluegel, W. (1995). Hydrological response units (HRUs) to preserve basin heterogeneity in hydrological modelling using PRMS/MMS- case study in the Bröl basin, Germany. *IAHS Publications-Series of Proceedings and Reports-Intern Assoc Hydrological Sciences*, **231**, 79–88. 14, 15

Goldberg, D. (1989). *Genetic Algorithms in search, optimization and machine learning*. Addison-Wesley Publishing Company. 31, 33, 42, 58, 75, 93, 95, B-1

Gupta, S. (1965). On some multiple decision (ranking and selection) rules. *Technometrics*, **7**, 225–245. 36

He, D., Lee, L.H., Chen, C.H., Fu, M.C. & Wasserkrug, S. (2010). Simulation optimization using the cross-entropy method with optimal computing

budget allocation. *ACM Transactions on Modeling and Computer Simulation (TOMACS)*, **20**, 4. 90

Juxin, L. (2012). *Optimal computing budget allocation for multi-objective simulation optimization*. Ph.D. thesis, National University of Singapore. 35, 73, 90, 91

Kim, S.H. & Nelson, B.L. (2001). A fully sequential procedure for indifference-zone selection in simulation. *ACM Transactions on Modeling and Computer Simulation (TOMACS)*, **11**, 251–273. 37

Kim, S.H. & Nelson, B.L. (2006). Chapter 17 Selecting the best system. In S.G. Henderson & B.L. Nelson, eds., *Simulation*, vol. 13 of *Handbooks in Operations Research and Management Science*, 501 – 534, Elsevier. 90

Kim, S.H. & Nelson, B.L. (2007). Recent advances in ranking and selection. In *Proceedings of the 39th Winter Simulation Conference: 40 years! The best is yet to come*, 162–172, IEEE Press. 35, 36, 89

Knowles, J., Thiele, L. & Zitzler, E. (2006). A Tutorial on the Performance Assessment of Stochastic Multiobjective Optimizers. TIK Report 214, Computer Engineering and Networks Laboratory (TIK), ETH Zurich. 76, 78

Krause, P. (2002). Quantifying the impact of land use changes on the water balance of large catchments using the J2000 model. *Physics and Chemistry of the Earth*, **27**, 663–673. 14, 15

Krause, P. & Kralisch, S. (2005). The hydrological modelling system J2000-knowledge core for JAMS. In *Proceedings of the 2005 International Congress on Modelling and Simulation*, 676–682. 14

Kroese, D.P. & Rubinstein, R.Y. (2005). The cross-entropy method for combinatorial optimization, rare event simulation and neural computation. *Annals of Operations Research*, **134(1)**. 8

Law, A.M. & Kelton, W.D. (2000). *Simulation modeling and analysis*. McGraw-Hill, 3rd edn. 2, 64

Lee, L.H., Chew, E.P., Teng, S. & Goldsman, D. (2004). Optimal computing budget allocation for multi-objective simulation models. In *Proceedings of the 2004 Winter Simulation Conference*, IEEE. 40, 41

Lee, L.H., Chew, E.P. & Teng, S. (2006). Integration of statistical selection with search mechanism for solving multi-objective simulation-optimization problems. In *Proceedings of the 2006 Winter Simulation Conference*, 294–303, IEEE. 39, 41, 90, 92

Lee, L.H., Chew, E.P., Teng, S. & Chen, Y. (2008). Multi-objective simulation-based evolutionary algorithm for an aircraft spare parts allocation problem. *European Journal of Operational Research*, **189**, 476–491. 3, 90

Lee, L.H., Chen, C., Chew, E.P., Li, J., Pujowidianto, N.A. & Zhang, S. (2010a). A review of optimal computing budget allocation algorithms for simulation optimization problem. *International Journal of Operations Research*, **7**, 19–31. 35

Lee, L.H., Chew, E.P., Teng, S. & Goldsman, D. (2010b). Finding the non-dominated pareto set for multi-objective simulation models. *IIE Transactions*, **42**, 656–674. 35, 40, 41, 44, 45, 46, 47, 48, 50, 51, 53, 54, 55, 56, 59

Mattila, V. & Virtanen, K. (2014). Ranking and selection for multiple performance measures using incomplete preference information. *European Journal of Operational Research*. 39, 40

Morrice, D.J., Butler, J. & Mullarkey, P.W. (1998). An approach to ranking and selection for multiple performance measures. In *Proceedings of the 30th Winter Simulation Conference*, 719–726, IEEE Computer Society Press. 39

Nelson, B.L., Swann, J., Goldsman, D. & Song, W. (2001). Simple procedures for selecting the best simulated system when the number of alternatives is large. *Operations Research*, **49**, 950–963. 36

Papadopoulos, C.T., Vidalis, M.J., OKelly, M.E. & Spinellis, D. (2009). The buffer allocation problem. In *Analysis and Design of Discrete Part Production Lines*, 131–159, Springer. 68, 69

Papadopoulos, H.T. & Heavey, C. (1996). Queueing theory in manufacturing systems analysis and design: A classification of models for production and transfer lines. *European Journal of Operational Research*, **92**, 1–27. 70

Rengasamy, P. (2006). World salinization with emphasis on australia. *Journal of Experimental Botany*, **57**, 1017–1023. 11

Richards, L. (1954). Diagnosis and improvement of saline and alkali soils. *Soil Science*, **78**, 154. 11

Rinott, Y. (1978). On two-stage selection procedures and related probability-inequalities. *Communications in Statistics-Theory and methods*, **7**, 799–811. 36, 39

Rubinstein, R. (1999). The cross-entropy method for combinatorial and continuous optimization. *Methodology and computing in applied probability*, **1**, 127–190. 7

Rubinstein, R.Y. & Kroese, D.P. (2004). *The Cross-Entropy Method: A Unified Approach to Combinatorial Optimization, Monte-Carlo Simulation, and Machine Learning*. Springer. 8, 10

Swisher, J.R., Jacobson, S.H. & Yücesan, E. (2003). Discrete-event simulation optimization using ranking, selection, and multiple comparison procedures: A survey. *ACM Transactions on Modeling and Computer Simulation (TOMACS)*, **13**, 134–154. 35, 36

Teng, S., Lee, L.H. & Chew, E.P. (2010). Integration of indifference-zone with multi-objective computing budget allocation. *European Journal of Operational Research*, **203**, 419–429. 40, 57, 58, 59, 60, 61

Weber, M. (1987). Decision making with incomplete information. *European Journal of Operational Research*, **28**, 44–57. 40

Yang, S., Wu, C. & Hu, S.J. (2000). Modeling and analysis of multi-stage transfer lines with unreliable machines and finite buffers. *Annals of Operations Research*, **93**, 405–421. 70

Appendix A

Matlab® code for the MOCBA algorithm

This appendix provides the Matlab® code for implementing the simple multi-objective optimal computing budget allocation algorithm by Chen & Lee (2010) to the inventory problem of Chapter 3.

```
function[Sp] = MOCBA_sSimple() % returns the Pareto set


 D = 186;    % Number of scenarios under investigation
 NT = 700;   % Total computing budget
 v = -1;     % Iteration index
 no = 5;     % Initial number of replications for each scenario
 N = no*ones(D,1);     % Vector of initial number of replications for
     scenarios
 Nt = sum(N); % Current simulation budget exhausted
 tau = 30;     % Max possible no. reps that can be allocated to a
    design at each iteration
```

```
delta = 200; % Increase the total number of simulation replications
    by a certain amount each iteration
DV = csvread('DV.csv'); % Read the decision variable values for the
    scenarios
I = horzcat(DV, N);
ticID = tic; % Start clock, not necessary

while Nt < NT %(termination conditioned is not satisfied)

csvwrite('DV_R.csv', I); % Input ready for C# (decision variable
   values and the numebr of replicatios)
system('bin\Release\RunExperimentsConsole.exe');   % Run the C#
   code (Simio model)

v = v + 1 % Increment iteration index

% Run the 'SortCResults.m' function
if v == 0
 S =  SortCResultsv(DV,N); % Input DV and reps, output results
    matrix: DV, scenario, Avg, Std dev, HW and reps
else
 S =  SortCResults(I,S, SS(2:size(SS,1),1), SN(2:size(SN,1),1)); %
    Input DV and reps, output Results matrix: DV, scenario, Avg,
    Std dev, HW and reps
end

% Run the Pareto Ranking algorithm
[Sp, Spn] = ParetoRanking(1, S); % Return the Sp(Elite set) and Spn
   (Non-Pareto)
```

```
%-------------------------------------------------------------------------
 % Determine kji
  K1 = zeros(D,D); K2 = zeros(D,D);     kji = zeros(D, D);
  for i = 1:D
      for j = 1:D
         K1(i,j) = ( (S(j,4) - S(i,4)) * abs(S(j,4) - S(i,4)) ) / (
             S(i,6)^2 + S(j,6)^2 );
         K2(i,j) = ( (S(j,5) - S(i,5)) * abs(S(j,5) - S(i,5)) ) / (
             S(i,7)^2 + S(j,7)^2 );
           if K1(i,j) > K2(i,j)
              kji(i,j) = 1;
           else kji(i,j) = 2;
           end
      end
  end

 % Determine ji
   J = 100*ones(D,D);
   for i = 1:D
       for j = 1:D
           if j~=i
               J(i,j) = ( (S(j,3 + kji(i,j)) - S(i,3 + kji(i,j))) *
                   abs(S(j,3 + kji(i,j)) - S(i,3 + kji(i,j))) ) / ( S
                   (i,5 + kji(i,j))^2 + S(j,5 + kji(i,j))^2 );
           end
       end
   end
   jii = min(J,[],2);
   ji = zeros(D,1);
   for u = 1:D
```

```
        for w = 1:D
            if J(u,w) == jii(u,1)
               ji(u,1) = w;
            end
        end
    end

%-------------------------------------------------------------------------
% Calculate SA set
    c1 = zeros(D,1); Sa = 0; Sb = 0;
     for h = 1:D
         c2 = 100*ones(D,1);
         c1(h,1) = (S(ji(h,1),3+kji(h,ji(h,1)))-S(h,3+kji(h,ji(h,1))
            ))^2/(S(h,5+kji(h,ji(h,1)))^2+ S(ji(h,1),5+kji(h,ji(h,1)
            ))^2);
         for i = 1:D
            if ji(i, 1) == h
               c2(i,1) = (S(h,3+kji(i,h))-S(i,3+kji(i,h)))^2/(S(i,5+
                  kji(i,h))^2+ S(h,5+kji(i,h))^2);
            end
         end
         c2m = min(c2);

         if c1(h,1) < c2m
                  Sa = vertcat(Sa, h);    % Recording what is in SA
             else Sb = vertcat(Sb, h);  % Recording what is not in
                SA therefore in SB
         end
      end
```

```
    % Full set of SA indexed to scenario number
    SA = Sa(2:size(Sa,1),1); % cut off first element of Sa(0).
    SB = Sb(2:size(Sb,1),1);

%-------------------------------------------------------------------
ah = zeros(size(SA,1),1);
ad = zeros(size(SB,1),1);
m = round(random('Uniform', 1, size(SA, 1)));  % Fixed index to a
    scenario in SA

% Calculate alphah
   for h = 1:size(SA,1) % h is an element of SA
    ah(h, 1) = ((S(SA(h,1),5+kji(SA(h,1),ji(SA(h,1),1)))/(S(ji(SA(h
        ,1),1),3+kji(SA(h,1),ji(SA(h,1),1))) - S(SA(h,1),3+kji(SA(h
        ,1),ji(SA(h,1),1))))  /  (S(SA(m,1),5+kji(SA(m,1),ji(SA(m
        ,1),1)))/(S(ji(SA(m,1),1),3+kji(SA(m,1),ji(SA(m,1),1)))-S(SA
        (m,1),3+kji(SA(m,1),ji(SA(m,1),1))))))^2;
   end

% Calculate alphad
    for d = 1:size(SB,1)    % d is an element of SB
       a = zeros(size(SA,1),1);
       for h = 1:size(SA,1)
           if ji(SA(h,1) ,1) == SB(d,1)
              a(h,1) = (S(SB(d,1),5+kji(SA(h,1),SB(d,1)))^2*ah(h,1)
                  ^2)/S(SA(h,1),5+kji(SA(h,1),SB(d,1)))^2;
           end
       end
     ad(d, 1) = sqrt(sum(a));  %sum and sqrt the i vector
    end
```

```
    B = sum(ah) + sum(ad); % sum for S
%-------------------------------------------------------------------------
 % Calculate alpha SA,SB order.
    alpha = zeros(D,1);
    for t = 1:size(SA,1)
        alpha(t,1) =  ah(t,1)/B;
    end
    for tt = 1:size(SB,1)
        alpha (tt+size(SA,1),1) = ad(tt,1)/B;
    end
    alphas = sortrows(horzcat(alpha, vertcat(SA,SB)),2); % scenario
       number order

    r = round(delta*alphas(1:D,1)); rS = horzcat(rS, r); % multiply
       the alpha allocation by delta
    reps = zeros(D, 1);
    for z = 1:D
        reps(z,1) = min(tau, r(z,1)); % calculate required reps.
           reps is in scenario numbers order (same order as N)
    end

    N = N + reps % Add previous reps and new reps
    Nt = sum(N) % Keep track of Nt. sum(N) = Nt + sum(reps). Nt +
       delta =/ Nt because of tau i.e. reps might be allocated but
       not run

    % Do not rerun sceanrio if no more replications were allocated
       to it.
    I = [0 0 0]; SS = 0; SN = 0;
```

```
    for w = 1:D
        if N(w,1) ~= S(w,10) % run again
            SS = vertcat(SS,w); % Stores the scenario number to run
               again
            I =  vertcat(I, horzcat(S(w,1:2), N(w,1))); % Stores the
                DV and N
        else SN = vertcat(SN, w); % scenario number not to run again
        end
    end
    I = I(2:size(I,1),1:3);

 end % ends while loop

elapsedtime = toc(ticID)/60
Pareto = size(Sp, 1)
end
```

Appendix B

Pareto ranking algorithm with indifference-zone

Algorithm 9 presents the Pareto ranking method of Goldberg (1989) which is adapted to incorporate an indifference-zone. It uses the same working matrix **W** as Table 3.1. The algorithm is put forth for two minimising objectives.

The algorithm was formed by combining the concept of Pareto dominance and ranking from Goldberg (1989) and the conditions in (3.62). The first condition is implemented in lines 2 to 19 of Algorithm 9 and the second condition in lines 20 to 38.

From experimentation in this study, it was found that the conditions of the algorithm are strict and scenarios are easily eliminated as contenders for the Pareto set. For that reason, care should be taken when choosing indifference-zones for objectives.

Algorithm 9 Pareto ranking algorithm with indifference-zone (Minimisation)

1: Input: Working matrix $\mathbf{W}$ with N rows and $(D+H+1)$ columns, indifference-zone values δ_k^* for objective $k \in H$ and user-selected threshold ρ_E.
2: Set $j = D + 1$.
3: Set $i = j + 1$.
4: Sort the working matrix $\mathbf{W}$ with the values in column j in descending order.
5: **for** $r_p = 1 \rightarrow N - 1$ **do**
6: **for** $r_q = r_p \rightarrow N - 1$ **do**
7: **if** $\mathbf{W}(r_p, D + H + 1) < \rho_E$ **then**
8: **if** $\mathbf{W}(r_p, i) - \delta_k^* \geq \mathbf{W}(r_q + 1, i)$ **then**
9: Increment the rank value in $\mathbf{W}(r_p, D + H + 1)$.
10: **end if**
11: **end if**
12: **end for**
13: **end for**
14: Increment j.
15: Decrement i.
16: **if** $k < H$, **then**
17: Return to Step 4
18: **else** continue with the rows in $\mathbf{W}$ with rank value not exceeding ρ_E as the temporary non-dominated vector.
19: **end if**
20: Let T = number of scenarios temporarily designated as non-dominated.
21: Set $j = D + 1$.
22: Set $i = j + 1$.

Algorithm 9 Pareto ranking algorithm with indifference-zone (continued)

23: Sort the working matrix $\mathbf{W}$ with the values in column j in descending order.
24: **for** $r_p = 1 \rightarrow T - 1$ **do**
25: **for** $r_q = r_p \rightarrow T - 1$ **do**
26: **if** $\mathbf{W}(r_p, D + H + 1) < H$ **then**
27: **if** $\mathbf{W}(r_p, i) + \delta_k^* > \mathbf{W}(r_q + 1, i)$ **then**
28: increment the rank value in $\mathbf{W}(r_p, D + H + 1)$.
29: **end if**
30: **end if**
31: **end for**
32: **end for**
33: Increment j.
34: Decrement i.
35: **if** $k < H$, **then**
36: return to Step 23
37: **else** return the rows in $\mathbf{W}$ with rank value not exceeding H as the non-dominated vector.
38: **end if**

Appendix C

Parameter analysis of multi-objective ranking and selection algorithms

In this appendix, the results of varying parameters for the MOCBA and MOCBA_IZ algorithms are documented. Analysing these results enables the researcher to choose good parameters for the experiments going forward.

The parameters tested for MOCBA and MOCBA_IZ using the buffer allocation problem are given in Tables C.1 and C.2 respectively.

Table C.3 and C.4 contain the parameters assessed for the MOCBA and MOCBA_IZ algorithm for the inventory problem.

Table C.1: MOCBA parameter analysis for the buffer allocation problem.

Experiment number	Parameters					Results				
	Parameter to vary	Nt	Δ	n_0	τ	Iterations	Final Nt	Run time (min)	No. in Pareto set	HA
1	Nt	2000	300	5	20	3	1915	1.934	74	2.2305
2		3000	300	5	20	11	2938	4.3001	72	2.9215
3		4000	300	5	20	16	3891	7.5236	84	2.9464
4		5000	300	5	20	25	4740	11.833	84	2.9591
5		10000	300	5	20	66	9993	59.82	89	2.5830
6	Δ	4000	200	5	20	23	3994	9.596	89	2.5659
7		4000	400	5	20	21	3976	9.3373	73	2.6650
8		4000	500	5	20	14	3981	7.7384	83	2.6500
9		4000	600	5	20	13	3916	7.5734	92	2.7646
10	n_0	5000	300	10	20	15	4907	8.8238	102	2.6874
11		4000	300	10	10	10	3677	5.6769	119	2.6848
12		8000	300	20	20	18	7995	17.436	106	2.5209
13		10000	300	30	20	14	9879	17.657	112	2.6916
14	τ	4000	300	5	10	28	3982	12.594	78	2.9087
15		4000	300	5	30	13	3769	6.1512	80	2.9523
16		4000	300	5	40	13	3930	7.6928	84	2.6656
17		4000	300	5	50	13	3866	5.7656	78	2.9614
18		4000	300	5	10000	9	3938	5.4022	80	2.5825

Table C.2: MOCBA_IZ parameter analysis for the buffer allocation problem.

	Parameters							Results				
Experiment number	Parameter to vary	Nt	Δ	n_0	τ	TPR IZ	WIP IZ	Iterations	Final Nt	Run time (min)	No. in Pareto set	HA
1	Nt	2000	300	5	20	0.005	0.5	2	1935	3.114	0	0
2		3000	300	5	20	0.005	0.5	5	2739	6.605	5	0.0898
3		4000	300	5	20	0.005	0.5	9	3829	13.99	4	0.3304
4		5000	300	5	20	0.005	0.5	13	4895	24.195	3	0.0518
5		10000	300	5	20	0.005	0.5	31	9758	112.06	3	0.0599
6	Δ	4000	200	5	20	0.005	0.5	15	3913	19.652	4	0.0646
7		4000	400	5	20	0.005	0.5	6	3625	10.07	4	0.0856
8		4000	500	5	20	0.005	0.5	5	3708	9.313	4	0.1002
9		4000	600	5	20	0.005	0.5	4	3637	7.953	6	0.0502
10	n_0	5000	300	10	20	0.005	0.5	8	4870	15.938	4	0.0501
11		5000	300	10	10	0.005	0.5	8	4813	15.501	5	0.0773
12		8000	300	20	20	0.005	0.5	9	7812	27.958	5	0.0927
13	τ	4000	300	5	10	0.005	0.5	9	3790	13.858	5	0.3609
14		4000	300	5	30	0.005	0.5	9	3845	14.113	4	0.3316
15		4000	300	5	40	0.005	0.5	9	3845	13.95	4	0.3316
16		4000	300	5	50	0.005	0.5	9	3845	14.089	4	0.3316
17		4000	300	5	10000	0.005	0.5	9	3845	14.071	4	0.3316
18	IZ	4000	300	5	20	0.001	1	18	3870	22.582	4	0.0529
19		4000	300	5	20	0.001	0.5	11	3982	12.298	7	0.1651
20		4000	300	5	20	0.001	0.1	9	3800	9.985	47	2.9933
21		4000	300	5	20	0.005	1	9	3764	13.203	2	0
22		4000	300	5	20	0.005	0.1	9	3929	15.567	25	2.8769
23		4000	300	5	20	0.01	1	9	3813	14.412	1	0
24		4000	300	5	20	0.01	0.5	9	3837	15.27	1	0
25		4000	300	5	20	0.01	0.1	9	3964	15.172	12	2.4721

Table C.3: MOCBA parameter analysis for the inventory model.

	Parameters					Results				
Experiment number	Parameter to vary	Nt	Δ	n_0	τ	Iterations	Final Nt	Run time (min)	No. in Pareto set	HA
1	Nt	1500	200	5	20	4	1444	3.88	50.00	192.01
2		2000	200	5	20	9	1994	6.24	44.00	181.68
3		3000	200	5	20	20	2952	12.89	49.00	178.46
4		4000	200	5	20	31	3976	27.31	49.00	176.45
5		5000	200	5	20	42	4979	41.76	53.00	174.20
6		10000	300	5	20	67	9976	194.40	59.00	102.01
7	Δ	3000	100	5	20	29	2951	15.75	48.00	175.37
8		3000	300	5	20	14	2891	13.28	50.00	182.73
9		3000	400	5	20	13	2891	10.96	43.00	165.57
10		3000	500	5	20	11	2848	10.53	50.00	101.38
11	n_0	5000	200	10	20	28	4914	33.88	61.00	203.33
17		4000	200	10	20	18	3865	18.39	57.00	204.11
18		3000	200	10	10	12	2982	12.38	49.00	116.45
19		3000	300	10	20	9	2962	9.90	54.00	137.24
20		6000	300	20	20	16	5877	31.849	62.00	118.94
21		10000	300	25	20	41	9838	139.42	62.00	107.02
22		10000	300	30	20	30	9843	110.16	67.00	107.98
12	τ	3000	200	5	10	29	2980	18.52	50.00	182.26
13		3000	200	5	30	16	2987	10.93	38.00	100.48
14		3000	200	5	40	12	2845	11.01	48.00	171.54
15		3000	200	5	50	12	2986	10.16	48.00	133.28
16		3000	200	5	10000	10	2878	7.99	36.00	172.54

Table C.4: MOCBA_IZ parameter analysis for the inventory model.

	Parameters							Results				
Experiment number	Parameter to vary	Nt	Δ	n_0	τ	SL IZ	Cost IZ	Iterations	Final Nt	Run time (min)	No. in Pareto set	HA
1	Nt	1500	200	5	20	0.01	5	3	1499	5.1064	22	78.611
2		2000	200	5	20	0.01	5	5	1873	8.5583	24	93.2561
3		3000	200	5	20	0.01	5	11	2994	24.679	23	93.0706
4		4000	200	5	20	0.01	5	16	3916	47.396	23	93.2104
5		5000	200	5	20	0.01	5	21	4862	75.822	23	93.2738
6		10000	300	5	20	0.01	5	31	9856	245.66	21	94.9126
7	Δ	3000	100	5	20	0.01	5	23	2931	39.309	22	171.436
8		3000	300	5	20	0.01	5	7	2953	18.436	22	95.4018
9		3000	400	5	20	0.01	5	5	2876	14.568	25	95.8343
10		3000	500	5	20	0.01	5	4	2838	12.451	25	95.7476
11	n_0	5000	200	10	20	0.01	5	16	4877	60.58	22	88.5763
12		3000	200	10	20	0.01	5	5	2818	14.628	22	86.1954
13		6000	200	20	20	0.01	5	12	5975	59.351	22	81.2324
14	τ	3000	200	5	10	0.01	5	11	2988	24.437	23	92.4163
15		3000	200	5	30	0.01	5	11	2994	24.625	23	93.0706
16		3000	200	5	40	0.01	5	11	2994	24.742	23	93.0706
17		3000	200	5	50	0.01	5	11	2994	24.681	23	93.0706
18		3000	200	5	10000	0.01	5	11	2994	24.518	23	93.0706
19	IZ	3000	200	5	20	0.05	10	11	2955	32.649	0	0
20		3000	200	5	20	0.05	5	11	2955	32.526	1	0
21		3000	200	5	20	0.05	1	11	2955	32.674	1	0
22		3000	200	5	20	0.01	10	11	2992	27.954	19	92.4397
23		3000	200	5	20	0.01	1	13	2875	20.574	27	96.2485
24		3000	200	5	20	0.001	10	11	2940	26.83	37	95.59
25		3000	200	5	20	0.001	5	11	2924	25.463	37	95.6925
26		3000	200	5	20	0.001	1	13	2974	22.619	38	100.079